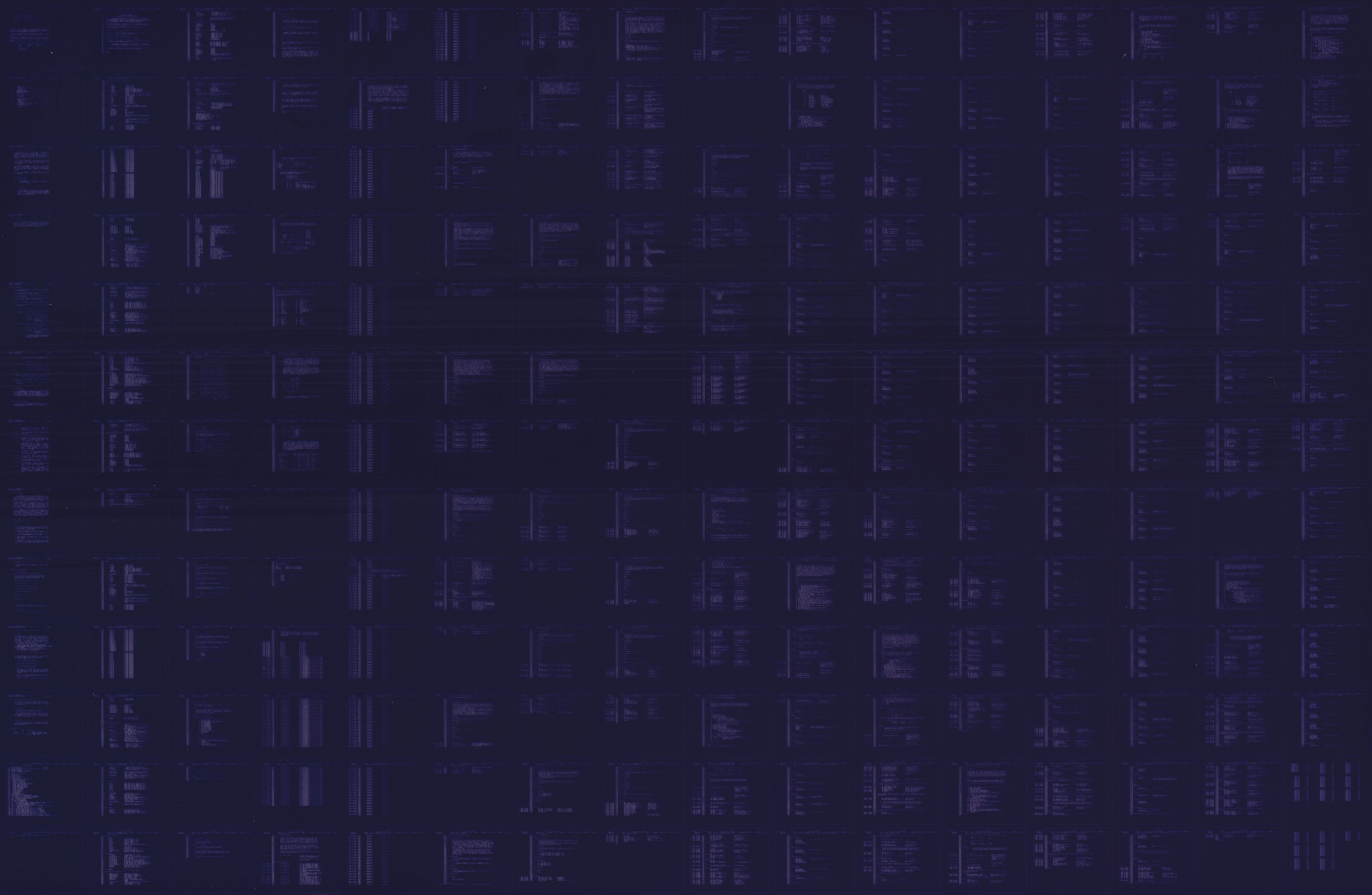
ENKCG-2.25 VAX 11730 IDC II
MICRO DIAG

COPYRIGHT (c) 1982-84
EP-ENKCG-DL-2.25
FICHE 01 OF 02

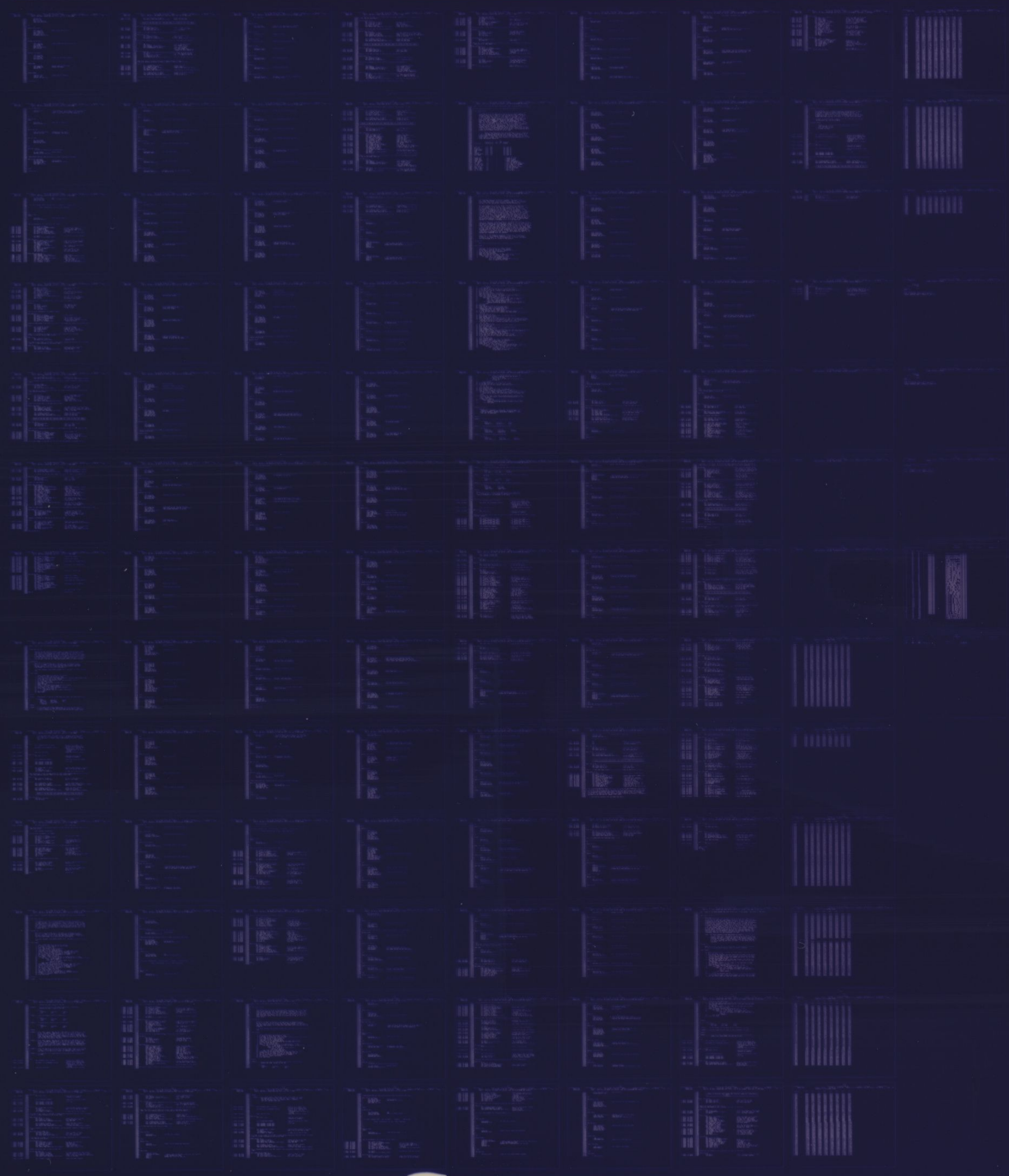
MAY 1984
digital
Made In USA



ENKCG-2.25 VAX 11730 IDC II
MICRO DIAG

COPYRIGHT (c) 1982-84
EP-ENKCG-DL-2.25
FICHE 02 OF 02

MAY 1984
digital
Made In USA



IDENTIFICATION

Product code: ZZ-ENKCG VERSION 2.25

Product name: MICRO-DIAGNOSTIC FOR VAX-11/730 IDC MODULE

Product date: 22-Feb-84

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1984 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC
DECUS
UNIBUS
DECsystem-10
MASSBUS
VAX
DECSYSTEM-20
PDP
VMS

```
 ¯ ¯ ¯ ¯ ¯ ¯ ¯
) ) ) ) ) ) ) )
)d)i)g)i)t)a)l)
)_)_)_)_)_)_)_)
```

Table of Contents

1.0 ABSTRACT 3
1.1 Product Users And Uses 3
1.1.1 Manufacturing 3
1.1.2 Field Service 3
1.1.3 Customer 4
2.0 HARDWARE REQUIREMENTS 5
3.0 SOFTWARE REQUIREMENTS 6
4.0 PREREQUISITES AND RESTRICTIONS 6
4.1 Restrictions 7

5.0 OPERATING INSTRUCTIONS 8
5.1 Event Flags 8
5.2 APT Setup 8

6.0 PROGRAM FUNCTIONAL DESCRIPTION 9
6.1 Program Overview 9
6.2 Program Size 9
6.3 Program Run Times 10
6.4 Error Reports 10
6.5 Performance During Hardware Failures 11
6.6 Program Applications 11
6.7 Test Descriptions 11

7.0 MAINTENANCE HISTORY 11

1 ABSTRACT

This program is a micro-diagnostic designed for manufacturing as an aid in the repair of NEBULA IDC (Integrated Disk Controller) modules, and in the field to function as a key part of the NEBULA CRD (Customer Runnable Diagnostic) package in assuring that there are no problems in booting the diagnostic supervisor from an IDC RL02 prior to passing control to the macro machine to execute the boot function.

IDC micro-diagnostic part I covered all IDC logic which did not require interaction with attached disks. Reference the IDC part I specification for further detail. IDC micro-diagnostic part II is aimed exclusively at the disk- IDC interface logic.

The IDC micro-diagnostic is an extension to the Nebula base machine micro-diagnostic package. That is, it runs under MICMON (micro monitor) control. The IDC micro-package part II will not be included in default DI testing. It can be invoked only through use of the Section options of the DIAGNOSE command.

The IDC micro-diagnostic will be packaged on the same TU58 tape that contains the MICMON and the base machine micro-diagnostic.

1.1 Product Users And Uses

1.1.1 Manufacturing -

The diagnostic will be used by Manufacturing for testing and repair of IDC modules. This diagnostic will be APT compatible.

1.1.2 Field Service -

The diagnostic will be available as an optional test for use by Field Service to isolate and repair faults on the IDC module associated with disk interface logic. It can be invoked by DI SE ENKCG. Reference section 3.1 for proper disk drive setup required in both CRD mode and NORMAL mode.

1.1.3 Customer -

This micro-diagnostic forms a key part of the NEBULA CRD package. It verifies that the required CRD RL02 diagnostic pack is mounted and that is accessible while the console processor is still in control of diagnostic test flow. This should help prevent problems occuring during the RL02 boot (i.e. the associated "uncontrolled" fault detection).

2 HARDWARE REQUIREMENTS

This program will run with the minimum hardware configuration of a WCS, CPU, MCT, IDC, 2 RL02's (or 1 RL02 and 1 R80) and an array module.

Maximum main memory allowed is 5 one meg boards, for a total of 5 megabytes of main memory storage.

Other options may be installed without affecting the diagnostic's performance.

This paragraph defines the requirements for execution of the diagnostic with respect to the following factors:

CPU Family - The diagnostic executes on a VAX/NEBULA processor only.

Disk Drives - Acceptable configurations for each mode as follows :

*** NORMAL or APT mode : ***

Configuration 1 : Drive 0 = R80, write enabled and spun up.

Drive 1 = RL02, write enabled

Drive 2 and 3 = don't care and spun up.

Pack Label = "SCRATCH" for NORMAL mode

Pack Label = "SCRATCH" or "don't care"
for APT mode
(reference to Restrictions below)

Configuration 2 : Drive 0 = RL02, state = don't care.

Drive 1 = RL02, write enabled and spun up.

Drive 2 and 3 = don't care

Pack Label = "SCRATCH" for NORMAL mode

Pack Label = "SCRATCH" or "don't care"
for APT mode
(reference to Restrictions below)

Note : For the RL02, write data will be done only on a scratch pack under normal mode and for the R80, write data will be done on the maintenance cylinders.

Note : Under APT mode, SCRATCH label test can be de-selected via APT software register.

*** CRD mode : ***

Configuration 1 : Drive 0 = don't care.

Drive 1 = RL02, W.E. = don't care, spun up.

Drive 2 and 3 = don't care

Pack Label = "CRDPACKXX" (XX = rev. number)

Load Device - The diagnostic is loaded into the console RAM and WCS through the TU58 serial line. If running under APT, loading is via the APT serial line.

Operator Interface - The operator interface is provided by Micmon.

APT Compatibility - Full compatibility for loading and monitoring.

3 SOFTWARE REQUIREMENTS

This diagnostic can be run either under the VAX-11/730 micro monitor (ENKAA) in a standalone environment or under the CRD micro-monitor (ENKAB). The micro-diagnostic is written into WCS RAM, wiping out any system micro-code that may have been resident. The IDC function micro codes are also located in the IDC control store. The micro-diagnostic monitor takes up the area where the console software is normally present.

4 PREREQUISITES AND RESTRICTIONS

The WCS, MCT and DAP modules are assumed to have been tested and known to be good before this diagnostic is run. The programs ENKBA through ENKBE, ENKCA through ENKCD will perform this testing.

4.1 Restrictions

Restriction:	The CPU must be fully operational before this diagnostic is executed. Also IDC micro-diagnostic part I must successfully execute prior to running part II.
Explanation:	Fault isolation may be invalid if the CPU is not fully operational prior to executing this diagnostic.
Restriction:	Micro tests should be run sequentially.
Explanation:	The diagnostic is structured on a building block approach. If the IDC requires 20 tests, test 1 should be run before test 2, test 2 before test 3 etc. Fault isolation may be invalid if the micro tests are not run sequentially.
Restriction:	Write testing for RL02's requires a "scratch" labeled pack on drive 1. Note : This is not a restriction for CRD nor APT modes : In CRD mode no write testing is done by IDC micro-diagnostic part II. Reference section 4.1 for further detail. And in APT mode, the "scratch" label test is de-selectable through the software register of the P-table.
Explanation:	This diagnostic checks for a scratch pack only on drive 1. If no scratch pack is present on drive 1, an error is reported and then all further testing is skipped.
Restriction:	Drive 0 must be an R80 or an RL02 for Normal or APT mode (i.e. there must be a disk drive attached to port 0). Drive 1 must be an RL02.
Explanation:	The drive configuration sizing test will need 2 drives hook up to the drive ports; if either one or both is absent, the test will fail.
Restriction:	The outermost and innermost cylinders of the RL02 disk must be free of bad sectors. This, however, is not a restriction for CRD.
Explanation:	This restriction will much help simplify the tests as less complexity of coding will be required and thus trouble-shooting in Manufacturing will be simpler. For CRD mode more flexibility is incorporated in the similar tests to accomodate correctable errors in the field.

5 OPERATING INSTRUCTIONS

This diagnostic will be executed automatically as part of the testing under CRD. It may also be executed under normal Micmon by typing DI SE ENKCG after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette. See the OVERVIEW MANUAL for more information.

Under regular Micmon (Normal and APT), test 6 - the "SCRATCH" label test will halt when the disk pack labelled "SCRATCH" is not found. If "CONTINUE" is entered upon error halt, as a prevention from accidentally erasing data on the disk through subsequent write data tests, the test will loop-on-error. This however could be bypassed by entering DI TE 7 or DI TE 7 CO if the operator does wish to do so.

This diagnostic has incorporated a utility program at the end as a software setup to help manufacturing eliminate cumbersome setups before it can bring IDC in synchronization with the RL02. To get access into this program, DI TE 11 has to be entered under normal Micmon upon MIC> prompt. To defeat Micmon timeout so as to synchronize the devices infinitely, SET LOOP has to entered upon MIC> under normal Micmon before getting into the program. To get out simply type "CTRL P".

5.1 Event Flags

Not applicable

5.2 APT Setup

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port. It sets a flag in the CPU local store which can be read by this diagnostic.

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

While creating a program to run this diagnostic under APT, the following parameters must be specified :

MEMORY SIZE IN 1/4 MB BLOCKS: Enter the size of memory in 1/4 MB increments. For example, enter "4" if the machine has 1 MB of memory.

HARDWARE BYTE: This byte is divided into two nibbles; the upper one is for the position of RL02 and the lower one is for the R80 drive position. Thus if drive 0 has a R80, drive 1 has a RL02, the hardware byte will be 21. Or if drive 0 and 1 both have the RL02's, it will then be 30. And so forth.

SOFTWARE BYTE: Enter 1 will deselect test 6 which reads the label of the RL02 disk and default value will run test 1 through E.

6 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

IDC part II tests the following interface logic :

-The IDC microcode which is located in PROM. The goal is to exercise all of the function codes (except format) provided in the microcode with the availablity of the associated disk drives namely RL02 and R80.

-RL02/R80 drivers and receivers

-R80 tag control MUX

-RL02/R80 status lines and associated logic

-RL02/R80 sector pulse and associated logic

-RL02/R80 drive command lines

-Status Clock

-R80 RD Clock

-R80 Servo Clock

6.2 Program Size

This program runs in WCS RAM memory and is approximately 14K bytes long (utilizing 2591 micro-words in WCS).

6.3 Program Run Times

This program will take approximately 1 minute to run including initialization, but excluding load time.

6.4 Error Reports

Hardware Failure - A hardware failure will cause an error report that identifies the failing sub-system(s) test number, section name, the expected and received data, the error mask, and an error number. Possible failing sub-systems are: IDC only; IDC/R80; IDC/RL02 or IDC/R80/RL02. The Diagnostic listing or separate document will describe the error in sufficient detail so that trouble-shooting can be accomplished. There are no plans to print out chips or additional information.

Under CRD mode, upon incorrect setup of the RL02 disk drive, the test will issue a warning message as follow:

```
    "PROPER SETUP REQUIRED - CHECK FOR RL02 DIAGNOSTIC PACK 'CRDPACK01'
INSERTED AND SPINNING, IN DRIVE DQA1. IF SETUP IS CORRECT, SUSPECT
RL02 DRIVE PROBLEM : - THEN TYPE <CR> TO CONTINUE"
```

Under Normal Micmon, with the exception of error 2 of test 6, the diagnostic will report standard errors as well as setup errors with the following header:

```
        SECT  TST ERR   EXP   REC  OTHER  MSK  MODULE
```

Under Normal Micmon, error 2 of test 6 will have the expected and received data (labels) printed in ASCII but less "OTHER MSK MODULE" with the following header:

```
        SECT  TST ERR   EXP   REC
```

1. SECT is the program name (ENKCG)

2. TST is the test number that failed.

3. ERR is the error number that failed.

4. EXP is the expected (correct) data.

5. REC is the received (actual) data.

6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.

7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.

8. MODULE is the suspected module that caused the failure.

6.5 Performance During Hardware Failures

A power fail will cause a rebooting of the system. Other failures, such as a timeout or a WCS parity error, will print an error message and return to the MIC> prompt. The operator can then issue other commands, including a continue if desired.

6.6 Program Applications

Customers may use this program to help verify the basic integrity of the IDC and the RL02 in the system.

This program can also be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing subsystem by field service or engineering.

6.7 Test Descriptions

See the actual tests in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

7 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
1-JULY-82	1.0	Initial release. (CRD tests only)
3-SEPT-82	2.0	Complete diagnostic release
16-JAN-84	2.25	TSU05 configuration contains no RL02 error will occur ignore it

```
;                                 Table of Contents
```

```
; 723    FIELD DEFINITIONS FOR MAIN MICRO WORD                  VER 00.08
; 1729   MACRO DEFINITIONS                                      VER 00.02
; 2385   FIELD DEFINTITIONS FOR DATA PATH CONTROL MICRO WORD    VER 00.01
; 2438   CONTROL ROM CONTENTS                                   VER 00.01
; 3188   DIAGNOSTIC MACROS AND DEFINITIONS
; 3885   COMMON LS ASSIGNMENTS (TO REGISTERS)
; 4039   Microinstruction Formats
; 4422   Tables
; 4553   2901 ALU Control Description
; 4657   DIAGNOSTIC MACROS
; 4685   TEST POINTERS
; 4839   DIAGNOSTIC POINTERS
; 4929   LS DATA SECTION
; 5320   PROGRAM SPECIFIC DATA SECTION (LS 80-F7)
; 5689   WCS SUBROUTINES FOR CONSOLE USE
; 5690     GENERATE TRANSFER DATA
; 5735     DEPOSIT MCT CSR ROUTINE
; 5801     EXAMINE MCT CSR ROUTINE
; 5888     DEPOSIT MCT TRANSLATION BUFFER ROUTINE
; 6004     EXAMINE TRANSLATION BUFFER ROUTINE
; 6067     DEPOSIT UNIBUS MAP ROUTINE
; 6165     EXAMINE UNIBUS MAP ROUTINE
; 6228     DEPOSIT MAIN MEMORY ROUTINE
; 6286     EXAMINE MAIN MEMORY ROUTINE
; 6348     EXAMINE IDC ROUTINES
; 6414     DEPOSIT IDC ROUTINES
; 6486     SAVE WR ROUTINE
; 6537     RESTORE WR ROUTINE
; 6588   WCS SUBROUTINES FOR CPU USE
; 6589     ERROR ROUTINE
; 6756   START TRANSFER ROUTINE
; 6855   SUBROUTINES FOR IDC MICRODIAGNOSTIC
; 6856    WAIT FOR CRDY TO SET ROUTINE (NON DATA TRANSFER)
; 6908    WAIT FOR DRDY TO SET ROUTINE (NON DATA TRANSFER)
; 6960   GET A CHARACTER FROM THE DATA BUFFER ROUTINE
; 7010   STORE A LABEL IN A LONGWORD ROUTINE
; 7094   SHIFT RL02 HEADER ROUTINE
; 7159   DAR FORMAT FOR SEEK ROUTINE
; 7216   RL02 SEEK AND VERIFY ROUTINE
; 7305   Test 1  Y BUS VERIFICATION TEST (M8388 IDC MODULE OR M8390 CPU MODULE)
; 7428   Test 2  IDC SIZING TEST (M8388 IDC MODULE OR M8390 CPU MODULE)
; 7581   Test 3  DISK DRIVE CONFIGURATION SIZING TEST
; 7795   Test 4  RL GET STATUS FUNCTION TEST (not run in CRD mode)
; 8179   Test 5  RL READ HEADER AND SEEK TESTS (run in all modes - except CRD mode)
; 8943   Test 6  RL READ DATA TEST (look for
; 9750   Test 7  RL WRITE DATA FUNCTION TEST (not run in CRD mode)
; 10558  Test 8  RL WRITE CHECK DATA TEST (not run in CRD mode)
; 11300  Test 9  R80 GET STATUS FUNCTION TEST (not run in CRD mode)
; 11640  Test A  R80 SECTOR COUNT SIGNAL TEST (not run in CRD mode)
; 11867  Test B  R80 READ HEADER AND SEEK TEST (not run in CRD mode)
; 12834  Test C  R80 INITIALIZATION SIGNAL TEST (not run in CRD mode)
; 12986  Test D  R80 WRITE AND READ DATA FUNCTION TEST (not run in CRD mode)
; 14459  Test E  R80 WRITE CHECK FUNCTION TEST (not run in CRD)
; 15329  Test F  TEST FOR THE RIGHT CRD VOLUME LABEL ON RL02(M8388 IDC MODULE)
; 17223  Test 10  RL READ HEADER AND SEEK TESTS (run in all modes - 2nd test in CRD mode)
```

; 17419 Test 11 IDC SYNCHRONIZATION WITH RL02 PROGRAM (Manual mode only)

```
;1       .TITLE  "ENKCG Micro-diagnostic for IDC module  Rev 02.01"
;2       ;
;3       ;
;4       ;                        COPYRIGHT (c) 1982 BY
;5       ;             DIGITAL EQUIPMENT CORPORATION, MAYNARD,
;6       ;              MASSACHUSETTS.  ALL RIGHTS RESERVED.
;7       ;
;8       ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY  BE  USED  AND  COPIED
;9       ; ONLY  IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
;10      ; OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES THEREOF
;11      ; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER  PERSON.  NO
;12      ; TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
;13      ;
;14      ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
;15      ; SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
;16      ;
;17      ; DIGITAL  ASSUMES  NO  RESPONSIBILITY  FOR  THE  USE OR RELIABILITY OF ITS
;18      ; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
;19      ;
;20
;21      ;++
;22      ; FACILITY:       VAX-11/730 MICRO-DIAGNOSTIC.
;23      ;
;24      ; ABSTRACT:       This micro-diagnostic tests the interface hardware of the IDC
;25      ;                 module in the VAX-11/730 processor.
;26      ;
;27      ; ENVIRONMENT:  ENKAA  Micro-diagnostic Monitor
;28      ;
;29      ; AUTHOR:         George F. Ho     15-June-82        VERSION 1.0
;30      ;
;31      ; MODIFIED BY:  George F. Ho     27-Aug-82         VERSION 2.0
;32      ;
;33      ;   * Version 2 has the addition of tests supporting Manufacturing, for detail
;34      ;     please refer to the ENKCG.DOC version 2.0
;35      ;
;36      ;   ****  This diagnostic is also used in the Customer Runable Diagnostic (CRD)
;37      ;           whose micmon must be updated to reflect changes in this diagnostic.
;38      ;
;39      ;               Darryl L. Ball  02-Feb-84         VERSION 2.1
;40      ;
;41      ;     Modified for running under CRD monitor ENKAB. When drive 1 is TSU05 the
;42      ;     monitor will interrupt with a pack label warning message.
;43      ;
;44      ;--
;45      .nolist
;3187    .LIST
```

```
.PAGE    "DIAGNOSTIC MACROS AND DEFINITIONS"

D.ADRS/=<15:9>,.VALIDITY=<D.VAL>,.DEFAULT=0


          SAV.WR0=1                    ; STORAGE FOR WR0
          SAV.WR1=2                    ; STORAGE FOR WR1
          SAV.WR2=3                    ; STORAGE FOR WR2
          SAV.WR3=4                    ; STORAGE FOR WR3
          T5.SUB=5                     ; RESERVED FOR COMMON SUBROUTINES
          T6.SUB=6                     ; RESERVED FOR COMMON SUBROUTINES
          T7.SUB=7                     ; RESERVED FOR COMMON SUBROUTINES
          TRANSFER.POINTER=7           ; POINTER FOR TRANSFER ROUTINE
          MEMORY.SIZE=7                ; STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
                                       ;  SIZING
          FPA.FOUND=7                  ; STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
          UBE.FOUND=7                  ; STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
          T8=8                         ; TEMP LOCATION 8
          T9=9                         ; TEMP LOCATION 9
          T10=0A                       ; TEMP LOCATION 10
          T11=0B                       ; TEMP LOCATION 11
          T12=0C                       ; TEMP LOCATION 12
          T13=0D                       ; TEMP LOCATION 13
          T14=0E                       ; TEMP LOCATION 14
          T15=0F                       ; TEMP LOCATION 15
          PC=10                        ; MACRO PROGRAM COUNTER

          WR.BACKUP=11                 ; BACKUP WR FOR MOV MEM.DATA TO WR[]
          MM.LASTADDR+1=12             ; STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
                                       ;  1 (FIRST NON-EXISTANT MEMORY ADDRESS)
          #FF=13                       ; MASK
          #FFFF=14                     ; MASK
          #FF000000=15                 ; MASK
          #FFFFFF00=16                 ; MASK
          CSR0.MASK=16                 ; MASK (FFFFFF00)
          CSR1.MASK=17                 ; MASK (01003FFF)
          CSR2.MASK=18                 ; MASK (7FFE3FFF)
          #FE7FFFFF=19                 ; MASK
          UBS.SET=1A                   ; CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
          UBS.DATA=1B                  ; MASK (7FFF8000) USED AS ERROR MASK FOR UBS
                                       ; DATA TEST

          ERR.CON.NA=1E                ; CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
                                       ; LOADING CONTROL AND STATUS REG IN INITIAL
                                       ; TESTS
          ERR.CON=1F                   ; CONSTANT OF 500(H) BITS 10 & 8 SET FOR
                                       ; LOADING CONTROL AND STATUS REG IN INITIAL
                                       ; TESTS

          #1=20                        ; PATTERN 00000001 (HEX)
          #2=21                        ; PATTERN 00000002
          #4=22                        ; PATTERN 00000004
          #8=23                        ; PATTERN 00000008
          #10=24                       ; PATTERN 00000010
          #20=25                       ; PATTERN 00000020
```

```
;3243          #40=26               ; PATTERN 00000040
;3244          #80=27               ; PATTERN 00000080
;3245          #100=28              ; PATTERN 00000100
;3246          #200=29              ; PATTERN 00000200
;3247          #400=2A              ; PATTERN 00000400
;3248          #800=2B              ; PATTERN 00000800
;3249          #1000=2C             ; PATTERN 00001000
;3250          #2000=2D             ; PATTERN 00002000
;3251          #4000=2E             ; PATTERN 00004000
;3252          #8000=2F             ; PATTERN 00008000
;3253          #10000=30            ; PATTERN 00010000
;3254          #20000=31            ; PATTERN 00020000
;3255          #40000=32            ; PATTERN 00040000
;3256          #80000=33            ; PATTERN 00080000
;3257          #100000=34           ; PATTERN 00100000
;3258          #200000=35           ; PATTERN 00200000
;3259          #400000=36           ; PATTERN 00400000
;3260          #800000=37           ; PATTERN 00800000
;3261          #1000000=38          ; PATTERN 01000000
;3262          #2000000=39          ; PATTERN 02000000
;3263          #4000000=3A          ; PATTERN 04000000
;3264          #8000000=3B          ; PATTERN 08000000
;3265          #10000000=3C         ; PATTERN 10000000
;3266          #20000000=3D         ; PATTERN 20000000
;3267          #40000000=3E         ; PATTERN 40000000
;3268          #80000000=3F         ; PATTERN 80000000
;3269
;3270          BIT0=20              ; PATTERN 00000001
;3271          BIT1=21              ; PATTERN 00000002
;3272          BIT2=22              ; PATTERN 00000004
;3273          BIT3=23              ; PATTERN 00000008
;3274          BIT4=24              ; PATTERN 00000010
;3275          BIT5=25              ; PATTERN 00000020
;3276          BIT6=26              ; PATTERN 00000040
;3277          BIT7=27              ; PATTERN 00000080
;3278          BIT8=28              ; PATTERN 00000100
;3279          BIT9=29              ; PATTERN 00000200
;3280          BIT10=2A             ; PATTERN 00000400
;3281          BIT11=2B             ; PATTERN 00000800
;3282          BIT12=2C             ; PATTERN 00001000
;3283          BIT13=2D             ; PATTERN 00002000
;3284          BIT14=2E             ; PATTERN 00004000
;3285          BIT15=2F             ; PATTERN 00008000
;3286          BIT16=30             ; PATTERN 00010000
;3287          BIT17=31             ; PATTERN 00020000
;3288          BIT18=32             ; PATTERN 00040000
;3289          BIT19=33             ; PATTERN 00080000
;3290          BIT20=34             ; PATTERN 00100000
;3291          BIT21=35             ; PATTERN 00200000
;3292          BIT22=36             ; PATTERN 00400000
;3293          BIT23=37             ; PATTERN 00800000
;3294          BIT24=38             ; PATTERN 01000000
;3295          BIT25=39             ; PATTERN 02000000
;3296          BIT26=3A             ; PATTERN 04000000
;3297          BIT27=3B             ; PATTERN 08000000
```

```
;3298           BIT28=3C                        ; PATTERN 10000000
;3299           BIT29=3D                        ; PATTERN 20000000
;3300           BIT30=3E                        ; PATTERN 40000000
;3301           BIT31=3F                        ; PATTERN 80000000
;3302
;3303   ;SPECIAL MNEMONICS
;3304
;3305
;3306           SETCSR.FO=3C                    ; CONSTANT 10000000
;3307           SETCSR.IE=26                    ; CONSTANT 40
;3308           SETCSR.CRDY=27                  ; CONSTANT 80
;3309           SETCSR.DS0=4E                   ; CONSTANT 0
;3310           SETCSR.DS1=28                   ; CONSTANT 100
;3311           SETCSR.DS2=29                   ; CONSTANT 200
;3312           SETCSR.SSEI=36                  ; CONSTANT 400000
;3313           SETCSR.SSFLG=37                 ; CONSTANT 800000
;3314           SETCSR.MAINT=39                 ; CONSTANT 2000000
;3315           SETCSR.R80=35                   ; CONSTANT 200000
;3316
;3317
;3318   ;CSR ADDRESS MNEMONICS
;3319
;3320           CSR0=4E                         ; ALL BITS CLEAR TO ACCESS CSR 0
;3321           CSR1=22                         ; BIT 2 SET TO ACCESS CSR 1
;3322           CSR2=23                         ; BIT 3 SET TO ACCESS CSR 2
;3323
;3324   ;CONTROL AND STATUS WORD BITS
;3325
;3326           WARNING.IDC=25                  ; PRINT WARNING MESSAGE FOR RL02 NOT PRESENT
;3327                                           ;  (BIT 5)
;3328           PRINT.UBE=26                    ; PRINT IF UBE PRESENT OR NOT PRESENT (BIT 6)
;3329                                           ;  (INDICATOR IN LS 7)
;3330           PRINT.R80=27                    ; PRINT IF R80 PRESENT OR NOT PRESENT AND ON
;3331                                           ; WHICH DRIVE (BIT 7).  INDICATOR IN LS 7,
;3332                                           ; DRIVE NUMBER IN LS 6.
;3333           ERROR=28                        ; TEST ERROR INDICATION (BIT 8)
;3334           SET.PA.ERR=29                   ; TELLS CONSOLE TO CREATE A PARITY ERROR AT WCS
;3335                                           ;  ADDRESS POINTED TO BY LS 7 (BIT 9)
;3336           CONSOLE.LOE=2A                  ; INDICATES CONSOLE DOES ERROR LOOPING (BIT 10)
;3337                                           ;  (FOR INITIAL TESTS)
;3338           PRINT.MEM.SIZE=2B               ; PRINT MEMORY SIZE IN 256K BLOCKS (BIT 11)
;3339                                           ;  (SIZE IN LS 7)
;3340           PRINT.FPA=2C                    ; PRINT IF FPA PRESENT OR NOT PRESENT (BIT 12)
;3341                                           ;  (INDICATOR IN LS 7)
;3342           XFER.DATA=2D                    ; INDICATES A DATA TRANSFER TO LS OR MM (BIT 13)
;3343           EOS=2E                          ; END OF SECTION (BIT 14)
;3344           BEGIN.TEST=2F                   ; BEGINNING OF TEST (BIT 15)
;3345           INTERUPT.EN=30                  ; ENABLE INTERUPT OR SIGNAL SPECIFIED BY BITS
;3346                                           ;  17 THRU 22 AND 28 THRU 30 (BIT 16)
;3347
;3348            CONSOLE.HALT=31                ;   CONSOLE HALT INTERRUPT (BIT 17)
;3349            PWR.FAIL=32                    ;   PWR FAIL INT (BIT 18)
;3350            INTERVAL.TIM=33                ;   INTRVL TIM INT (BIT 19)
;3351            CONSOLE.ATTN=34                ;   CONSOLE ATTN INTERRUPT (BIT 20)
;3352            CONSOLE.ACK=35                 ;   CONSOLE ACK INTERRUPT (BIT 21)
```

```
;3353            DISABLE.MEM.REF=36      ;   DISABLE MEMORY REFERANCES (BIT 22)
;3354            CINIT=3C                ;   UNIBUS INIT SIGNAL TO THE MCT (BIT 28)
;3355            UBS.DCLO=3D             ;   UNIBUS DC LO INDICATION TO THE MCT (BIT 29)
;3356            UBS.BBSY=3E             ;   UNIBUS BUS BUSY INDICATOR TO MCT (BIT 30)
;3357
;3358           NA.EXP.REC=37            ; PRINT N/A UNDER EXPECTED AND RECEIVED (BIT 23)
;3359           OTHER.DATA=38            ; PRINT A VALUE UNDER OTHER DATA (BIT 24)
;3360           CONSOLE.LOOP=39          ; CONSOLE PERFORMS A LOOP ACCORDING TO LOOP
;3361                                    ; COUNT IN LS (BIT 25)
;3362           PRINT.CRD=3A             ; PRINT NUMBER OF SINGLE BIT ERRORS (BIT 26)
;3363           CLR.PA.ERR=3B            ; TELLS CONSOLE TO CLEAR PARITY ERROR AT WCS
;3364                                    ;  ADDRESS POINTED TO BY LS 7 (BIT 27)
;3365           PRINT.IDC=3F             ; PRINT IF IDC PRESENT OR NOT PRESENT (BIT 31)
;3366                                    ;  (INDICATOR IN LS 7)
;3367
;3368   ;MODULE CODES
;3369
;3370           WCS=20                   ; MODULE CODE FOR WCS BOARD (BIT 0 SET)
;3371           CPU=21                   ; MODULE CODE FOR CPU BOARD (BIT 1 SET)
;3372           MCT=22                   ; MODULE CODE FOR MCT BOARD (BIT 2 SET)
;3373           FPA=23                   ; MODULE CODE FOR FPA BOARD (BIT 3 SET)
;3374           ARRAY=24                 ; MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
;3375           IDC=25                   ; MODULE CODE FOR IDC BOARD (BIT 5 SET)
;3376           RL02=26                  ; DEVICE CODE FOR RL02 DISK SUBSYSTEM (BIT 6)
;3377           R80=27                   ; DEVICE CODE FOR R80 DISK SUBSYSTEM (BIT 7)
;3378
;3379   ;CSR 1 ERROR BITS
;3380
;3381           VALID.ERR=2E             ; VALID NOT SET IF 1 (BIT 14)
;3382           TB.PAR.ERR=2F            ; TB PARITY ERROR (BIT 15)
;3383           NXM=30                   ; NON-EXISTANT MEMORY ERROR (BIT 16)
;3384           UB.BSY=31                ; UNIBUS BUSY (BIT 17)
;3385           ADP.REG=32               ; UNIBUS ADAPTER REGISTER SELECT (BIT 18)
;3386           WR.ACROSS.PG=33          ; WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
;3387           OP.ERR=34                ; OPERATION ERROR (BIT 20)
;3388           TB.MISS=35               ; TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
;3389                                    ;   BUFFER) (BIT 21)
;3390           ACCESS.REFUSED=36        ; PROTECTION ERROR ON ACCESS (BIT 22)
;3391           MODIFY.REFUSED=37        ; PROTECTION ERROR ON WRITE (BIT 23)
;3392
;3393           CRD=3E                   ; SINGLE BIT ERROR CORRECTED (BIT 30)
;3394           RDS=3F                   ; UNCORRECTABLE DATA ERROR (BIT 31)
;3395
;3396
;3397   ;CSR 1 CONTROL BITS
;3398
;3399           ECC.DIS=39               ; CSR1 ECC DISABLE BIT (BIT 25)
;3400           DIAG.CHK=3A              ; CSR1 DIAG CHK BIT (BIT 26)
;3401           MME=3B                   ; CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
;3402           INH.CRD=3C               ; CSR1 INHIBIT REPORT CRD BIT (BIT 28)
;3403           TB.PAR.DIAG=3D           ; CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
;3404                                    ;  (BIT 29)
;3405
;3406
;3407   ;UBE CONTROL REGISTER BITS
```

```

        ;CONTROL REG 1
               GO=20                       ; START UBE (BIT 0)
               BR4=21                      ; ISSUE BUS REQUEST 4 (BIT 1)
               BR5=22                      ; ISSUE BUS REQUEST 5 (BIT 2)
               BR6=23                      ; ISSUE BUS REQUEST 6 (BIT 3)
               BR7=24                      ; ISSUE BUS REQUEST 7 (BIT 4)
               NPR=25                      ; ISSUE NPR (BIT 5)
               INT.ODNE=26                 ; INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
               RDY=27                      ; READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
               CO0=28                      ; C0 BIT (BIT 8)
               CO1=29                      ; C1 BIT (BIT 9)
               FUNA=2A                     ; FUNCTION BIT A (BIT 10
               FUNB=2B                     ; FUNCTION BIT B (BIT 11)
               CC+DAT=2C                   ; DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
               INH.DATIP.ROL=2D            ; INHIBIT ROL ON DATIP (BIT 13)
               DATOB.ON.DATIP=2E           ; DO DATOB AFTER DATIP CYCLE (BIT 14)
               ERR=2F                      ; EXERCISOR OR UNIBUS ERROR (BIT 15)

        ;CONTROL REG 2
               EXT.MEM0=20                 ; EXTENDED MEMORY BIT 0 (BIT 0)
               EXT.MEM1=21                 ; EXTENDED MEMORY BIT 1 (BIT 1)
               INH.INC.ADDR&CC=22          ; INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)
               INH.SACK=23                 ; INHIBIT ASSERING BUS SACK ON BUS GRANT (BIT 3)
               PWR.DWN.SEQ=24              ; ASSERT ACLO (BIT 4)
               WNG.GNT.BCK=25              ; NO BUS GRANT RECV TO HIGEST REQUEST (BIT 5)
               MAX.LATE.ERR=26             ; NPR AND BR LATENCY TIME EXCEEDED (BIT 6)
               NO.SACK.ERR=27              ; NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)
               NO.SSYN.ERR=28              ; NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)
               WRG.A.LINES=29              ; ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)
               NO.GNT+NOT.ONE.GNT=2A       ; NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)
               INTR.SSYN.ERR=2B            ; NO SSYN RECEIVED AFTER BUS INTR (BIT 11)
               ENB.PB=2C                   ; ENABLE PARITY BIT B (BIT 12)
               CCOVF=2D                    ; CYCLE COUNT OVERFLOW (BIT 13)
               TM.DLY=2E                   ; REQUEST BUS EVERY 10 USEC (BIT 14)

      ;BG6 MNEMONIC
               BG6=23                      ; BIT 3 SET FOR BG 6 ADDRESS

      ;ERROR AND CONTROL INFORMATION

               CONTROL.STATUS=40           ; CONTROL AND STATUS WORD
               ERROR.NUMBER=41             ; ERROR NUMBER OF CURRENT TEST
               EXPECTED.DATA=42            ; EXPECTED RESULT OF CURRENT TEST
               RECEIVED.DATA=43            ; ACTUAL RESULT OF CURRENT TEST
               ADDRESS.DATA=44             ; OTHER PERTINENT DATA
               ERROR.MASK=45               ; BITS TO CHECK IN RESULT
               MODULE.NUM=46               ; STORAGE OF MODULE NUMBER (CODED)
               LOOP.CNT=47                 ; STORAGE OF LOOP COUNT FOR CONSOLE LOOP
                                           ; CONTROL.
               ERROR.CONTROL=48            ; STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)
                LOE=20                     ;  LOOP ON ERROR IF SET (BIT0)
                NER=21                     ;  NO ERROR REPORT IF SET (BIT1)
                BELL=22                    ;  BELL ON ERROR IF SET (BIT2)
                HOE=23                     ;  HALT ON ERROR IF SET (BIT3)
```

```
;3463            SER=24                  ;  ENABLE ERROR REPORT ON CRD ERRORS IF SET
;3464            APT.PRESENT=25          ;  APT PRESENT IF SET (BIT5)
;3465            LOOP.COMMAND=26         ;  LOOP COMMAND TYPED IF SET (BIT 6)
;3466            SPECIAL=27              ;  SPECIAL BIT FOR LOOP ON SIZE TESTS (BIT 7)
;3467
;3468           APT.PARAM=49             ;APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-
;3469                                    ; WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1
;3470                                    ; AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)
;3471           APT.RESERVED=4A          ;RESERVED FOR FUTURE APT USE
;3472
;3473    ;MISCELLANEOUS CONSTANTS AND STORAGE
;3474
;3475           #AAAAAAAA=4C             ; CONSTANT
;3476           ALTER.ODD=4C             ; CONSTANT
;3477           #55555555=4D             ; CONSTANT
;3478           ALTER.EVEN=4D            ; CONSTANT
;3479           #0=4E                    ; CONSTANT
;3480           ZERO=4E                  ; CONSTANT
;3481           #FFFFFFFF=4F             ; CONSTANT
;3482           ONES=4F                  ; CONSTANT
;3483           PREVIOUS.ERROR=50        ; ERROR NUMBER OF LAST ERROR
;3484
;3485           DATA.1=51                ; STORAGE FOR FPA DATA
;3486           DATA.2=52                ; STORAGE FOR FPA DATA
;3487           DATA.3=53                ; STORAGE FOR FPA DATA
;3488           FIRST.FLT=54             ; STORAGE FOR FPA DATA
;3489           SEC.FLT=55               ; STORAGE FOR FPA DATA
;3490           THIRD.FLT=56             ; STORAGE FOR FPA DATA
;3491           T57=57                   ; TEMP LOCATION 57
;3492           T58=58                   ; TEMP LOCATION 58
;3493           T59=59                   ; TEMP LOCATION 59
;3494
;3495           BEDB=5A                  ;UBE (BUS EXERCISOR) DATA BUF REG
;3496           BECC=5B                  ;UBE (BUS EXERCISOR) CYCLE COUNT REG
;3497           BEBA=5C                  ;UBE (BUS EXERCISOR) ADDR REG
;3498           BECR1=5D                 ;UBE (BUS EXERCISOR) CONTROL REG 1
;3499           CLEAR.ERR.ADDR=5E        ;UBE CLEAR ERROR REG ADDRESS
;3500           BECR2=5F                 ;UBE (BUS EXERCISOR) CONTROL REG 2
;3501
;3502           #3(H)=60                 ; CONSTANT
;3503           #3=60                    ; CONSTANT
;3504           #12(H)=61                ; CONSTANT
;3505           #33333333=62             ; CONSTANT
;3506           #0F0F0F0F=63             ; CONSTANT
;3507           #00FF00FF=64             ; CONSTANT
;3508           #162(H)=65               ; CONSTANT FOR LS TRANSFER POINTER
;3509           BR7.IDENT=66             ; BR7 IDENTIFIER (1010(B) IN BITS 5-2)
;3510           BG7=67                   ; BG7 ADDRESS (BITS 2 AND 3 SET)
;3511
;3512    ;TEMPS FOR CPU TESTS
;3513           L30=30                   ;LS 30 TEMP (RESTORED TO 10000 AFTER USE)
;3514           L31=31                   ;LS 31 TEMP (RESTORE TO 20000 AFTER USE)
;3515           L53=53                   ;LS 53 TEMP
;3516           L73=73                   ;LS 73 TEMP
;3517
```

```
;3518  ;REGISTER ADDRESSES
;3519
;3520          CONTROL.OS=78              ; HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
;3521          SHIFT.OS(3-0)=79           ; 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
;3522                                     ;  (LS 20-2F)
;3523          SHIFT.OS(4-0)=7B           ; 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
;3524                                     ;  (LS 20-3F)
;3525          OS=7C                      ; OS REGISTER
;3526          DEC.CON=7D                 ; DECIMAL CARRIES
;3527          SIZE=7E                    ; SIZE REGISTER
;3528          MDT= 7E                    ; MEMORY REFERENCE DATA SIZE
;3529          INTERRUPT.VEC=7E           ; HARDWARE INTERRUPT VECTOR
;3530  ;
;3531  ;       THIS WORD IS THE HARDWARE CONDITION CODES.  THE CC'S CAN BE READ
;3532  ;       OR ARE WRITTEN HERE.  NO OTHER BITS IN THE PSL ARE EFFECTED.
;3533  ;
;3534          PSL.CC=7F                  ; PSL CONDITION CODES
```

```
;3535  .PAGE
;3536  XD.ADRS/=<16:9>,.VALIDITY=<XD.VAL>
;3537
;3538
;3539          SAV.WR0=1                ; STORAGE FOR WR0
;3540          SAV.WR1=2                ; STORAGE FOR WR1
;3541          SAV.WR2=3                ; STORAGE FOR WR2
;3542          SAV.WR3=4                ; STORAGE FOR WR3
;3543          T5.SUB=5                 ; RESERVED FOR COMMON SUBROUTINES
;3544          T6.SUB=6                 ; RESERVED FOR COMMON SUBROUTINES
;3545          T7.SUB=7                 ; RESERVED FOR COMMON SUBROUTINES
;3546          TRANSFER.POINTER=7       ; POINTER FOR TRANSFER ROUTINE
;3547          MEMORY.SIZE=7            ; STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
;3548                                   ;  SIZING
;3549          FPA.FOUND=7              ; STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
;3550          UBE.FOUND=7              ; STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
;3551          T8=8                     ; TEMP LOCATION 8
;3552          T9=9                     ; TEMP LOCATION 9
;3553          T10=0A                   ; TEMP LOCATION 10
;3554          T11=0B                   ; TEMP LOCATION 11
;3555          T12=0C                   ; TEMP LOCATION 12
;3556          T13=0D                   ; TEMP LOCATION 13
;3557          T14=0E                   ; TEMP LOCATION 14
;3558          T15=0F                   ; TEMP LOCATION 15
;3559          PC=10                    ; MACRO PROGRAM COUNTER
;3560
;3561          WR.BACKUP=11             ; BACKUP WR FOR MOV MEM.DATA TO WR[]
;3562          MM.LASTADDR+1=12         ; STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
;3563                                   ;  1 (FIRST NON-EXISTANT MEMORY ADDRESS)
;3564          #FF=13                   ; MASK
;3565          #FFFF=14                 ; MASK
;3566          #FF000000=15             ; MASK
;3567          #FFFFFF00=16             ; MASK
;3568          CSR0.MASK=16             ; MASK (FFFFFF00)
;3569          CSR1.MASK=17             ; MASK (01003FFF)
;3570          CSR2.MASK=18             ; MASK (FFFE3FFF)
;3571          #FE7FFFFF=19             ; MASK
;3572          UBS.SET=1A               ; CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
;3573          UBS.DATA=1B              ; MASK (7FFF8000) USED AS ERROR MASK FOR UBS
;3574                                   ; DATA TEST
;3575
;3576          ERR.CON.NA=1E            ; CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
;3577                                   ; LOADING CONTROL AND STATUS REG IN INITIAL
;3578                                   ; TESTS
;3579          ERR.CON=1F               ; CONSTANT OF 500(H) BITS 10 & 8 SET FOR
;3580                                   ; LOADING CONTROL AND STATUS REG IN INITIAL
;3581                                   ; TESTS
;3582
;3583          #1=20                    ; PATTERN 00000001 (HEX)
;3584          #2=21                    ; PATTERN 00000002
;3585          #4=22                    ; PATTERN 00000004
;3586          #8=23                    ; PATTERN 00000008
;3587          #10=24                   ; PATTERN 00000010
;3588          #20=25                   ; PATTERN 00000020
;3589          #40=26                   ; PATTERN 00000040
```

```
;3590          #80=27                ; PATTERN 00000080
;3591          #100=28               ; PATTERN 00000100
;3592          #200=29               ; PATTERN 00000200
;3593          #400=2A               ; PATTERN 00000400
;3594          #800=2B               ; PATTERN 00000800
;3595          #1000=2C              ; PATTERN 00001000
;3596          #2000=2D              ; PATTERN 00002000
;3597          #4000=2E              ; PATTERN 00004000
;3598          #8000=2F              ; PATTERN 00008000
;3599          #10000=30             ; PATTERN 00010000
;3600          #20000=31             ; PATTERN 00020000
;3601          #40000=32             ; PATTERN 00040000
;3602          #80000=33             ; PATTERN 00080000
;3603          #100000=34            ; PATTERN 00100000
;3604          #200000=35            ; PATTERN 00200000
;3605          #400000=36            ; PATTERN 00400000
;3606          #800000=37            ; PATTERN 00800000
;3607          #1000000=38           ; PATTERN 01000000
;3608          #2000000=39           ; PATTERN 02000000
;3609          #4000000=3A           ; PATTERN 04000000
;3610          #8000000=3B           ; PATTERN 08000000
;3611          #10000000=3C          ; PATTERN 10000000
;3612          #20000000=3D          ; PATTERN 20000000
;3613          #40000000=3E          ; PATTERN 40000000
;3614          #80000000=3F          ; PATTERN 80000000
;3615
;3616          BIT0=20               ; PATTERN 00000001
;3617          BIT1=21               ; PATTERN 00000002
;3618          BIT2=22               ; PATTERN 00000004
;3619          BIT3=23               ; PATTERN 00000008
;3620          BIT4=24               ; PATTERN 00000010
;3621          BIT5=25               ; PATTERN 00000020
;3622          BIT6=26               ; PATTERN 00000040
;3623          BIT7=27               ; PATTERN 00000080
;3624          BIT8=28               ; PATTERN 00000100
;3625          BIT9=29               ; PATTERN 00000200
;3626          BIT10=2A              ; PATTERN 00000400
;3627          BIT11=2B              ; PATTERN 00000800
;3628          BIT12=2C              ; PATTERN 00001000
;3629          BIT13=2D              ; PATTERN 00002000
;3630          BIT14=2E              ; PATTERN 00004000
;3631          BIT15=2F              ; PATTERN 00008000
;3632          BIT16=30              ; PATTERN 00010000
;3633          BIT17=31              ; PATTERN 00020000
;3634          BIT18=32              ; PATTERN 00040000
;3635          BIT19=33              ; PATTERN 00080000
;3636          BIT20=34              ; PATTERN 00100000
;3637          BIT21=35              ; PATTERN 00200000
;3638          BIT22=36              ; PATTERN 00400000
;3639          BIT23=37              ; PATTERN 00800000
;3640          BIT24=38              ; PATTERN 01000000
;3641          BIT25=39              ; PATTERN 02000000
;3642          BIT26=3A              ; PATTERN 04000000
;3643          BIT27=3B              ; PATTERN 08000000
;3644          BIT28=3C              ; PATTERN 10000000
```

```
                BIT29=3D                ; PATTERN 20000000
                BIT30=3E                ; PATTERN 40000000
                BIT31=3F                ; PATTERN 80000000

        ;SPECIAL MNEMONICS


                SETCSR.FO=3C            ; CONSTANT 10000000
                SETCSR.IE=26            ; CONSTANT 40
                SETCSR.CRDY=27          ; CONSTANT 80
                SETCSR.DSO=4E           ; CONSTANT 0
                SETCSR.DS1=28           ; CONSTANT 100
                SETCSR.DS2=29           ; CONSTANT 200
                SETCSR.SSEI=36          ; CONSTANT 400000
                SETCSR.SSFLG=37         ; CONSTANT 800000
                SETCSR.MAINT=39         ; CONSTANT 2000000
                SETCSR.R80=35           ; CONSTANT 200000


        ;CSR ADDRESS MNEMONICS

                CSRO=4E                 ; ALL BITS CLEAR TO ACCESS CSR 0
                CSR1=22                 ; BIT 2 SET TO ACCESS CSR 1
                CSR2=23                 ; BIT 3 SET TO ACCESS CSR 2


        ;CONTROL AND STATUS WORD BITS

                WARNING.IDC=25          ; PRINT WARNING MESSAGE FOR RL02 NOT PRESENT
                                        ;  (BIT 5)
                PRINT.UBE=26            ; PRINT IF UBE PRESENT OR NOT PRESENT (BIT 6)
                                        ;  (INDICATOR IN LS 7)
                PRINT.R80=27            ; PRINT IF R80 PRESENT OR NOT PRESENT AND ON
                                        ; WHICH DRIVE (BIT 7).  INDICATOR IN LS 7,
                                        ; DRIVE NUMBER IN LS 6.
                ERROR=28                ; TEST ERROR INDICATION (BIT 8)
                SET.PA.ERR=29           ; TELLS CONSOLE TO CREATE A PARITY ERROR AT WCS
                                        ;  ADDRESS POINTED TO BY LS 7 (BIT 9)
                CONSOLE.LOE=2A          ; INDICATES CONSOLE DOES ERROR LOOPING (BIT 10)
                                        ;  (FOR INITIAL TESTS)
                PRINT.MEM.SIZE=2B       ; PRINT MEMORY SIZE IN 256K BLOCKS (BIT 11)
                                        ;  (SIZE IN LS 7)
                PRINT.FPA=2C            ; PRINT IF FPA PRESENT OR NOT PRESENT (BIT 12)
                                        ;  (INDICATOR IN LS 7)
                XFER.DATA=2D            ; INDICATES A DATA TRANSFER TO LS OR MM (BIT 13)
                EOS=2E                  ; END OF SECTION (BIT 14)
                BEGIN.TEST=2F           ; BEGINNING OF TEST (BIT 15)
                INTERUPT.EN=30          ; ENABLE INTERUPT OR SIGNAL SPECIFIED BY BITS
                                        ;  17 THRU 22 AND 28 THRU 30 (BIT 16)

                  CONSOLE.HALT=31       ;   CONSOLE HALT INTERRUPT (BIT 17)
                  PWR.FAIL=32           ;   PWR FAIL INT (BIT 18)
                  INTERVAL.TIM=33       ;   INTRVL TIM INT (BIT 19)
                  CONSOLE.ATTN=34       ;   CONSOLE ATTN INTERRUPT (BIT 20)
                  CONSOLE.ACK=35        ;   CONSOLE ACK INTERRUPT (BIT 21
                 DISABLE.MEM.REF=36      ;   DISABLE MEMORY REFERANCES (BIT 22)
                 CINIT=3C                ;   UNIBUS INIT SIGNAL TO THE MCT (BIT 28)
                 UBS.DCLO=3D             ;   UNIBUS DC LO INDICATION TO THE MCT (BIT 29)
                 UBS.BBSY=3E             ;   UNIBUS BUS BUSY INDICATOR TO MCT (BIT 30)

                NA.EXP.REC=37            ; PRINT N/A UNDER EXPECTED AND RECEIVED (BIT 23)
                OTHER.DATA=38            ; PRINT A VALUE UNDER OTHER DATA (BIT 24)
                CONSOLE.LOOP=39          ; CONSOLE PERFORMS A LOOP ACCORDING TO LOOP
                                         ; COUNT IN LS (BIT 25)
                PRINT.CRD=3A             ; PRINT NUMBER OF SINGLE BIT ERRORS (BIT 26)
                CLR.PA.ERR=3B            ; TELLS CONSOLE TO CLEAR PARITY ERROR AT WCS
                                         ;  ADDRESS POINTED TO BY LS 7 (BIT 27)
                PRINT.IDC=3F             ; PRINT IF IDC PRESENT OR NOT PRESENT (BIT 31)
                                         ;  (INDICATOR IN LS 7)


        ;MODULE CODES

                WCS=20                   ; MODULE CODE FOR WCS BOARD (BIT 0 SET)
                CPU=21                   ; MODULE CODE FOR CPU BOARD (BIT 1 SET)
                MCT=22                   ; MODULE CODE FOR MCT BOARD (BIT 2 SET)
                FPA=23                   ; MODULE CODE FOR FPA BOARD (BIT 3 SET)
                ARRAY=24                 ; MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
                IDC=25                   ; MODULE CODE FOR IDC BOARD (BIT 5 SET)
                RL02=26                  ; DEVICE CODE FOR RL02 DISK SUBSYSTEM (BIT 6)
                R80=27                   ; DEVICE CODE FOR R80 DISK SUBSYSTEM (BIT 7)

        ;CSR 1 ERROR BITS

                VALID.ERR=2E             ; VALID NOT SET IF 1 (BIT 14)
                TB.PAR.ERR=2F            ; TB PARITY ERROR (BIT 15)
                NXM=30                   ; NON-EXISTANT MEMORY ERROR (BIT 16)
                UB.BSY=31                ; UNIBUS BUSY (BIT 17)
                ADP.REG=32               ; UNIBUS ADAPTER REGISTER SELECT (BIT 18)
                WR.ACROSS.PG=33          ; WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
                OP.ERR=34                ; OPERATION ERROR (BIT 20)
                TB.MISS=35               ; TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
                                         ;   BUFFER) (BIT 21)
                ACCESS.REFUSED=36        ; PROTECTION ERROR ON ACCESS (BIT 22)
                MODIFY.REFUSED=37        ; PROTECTION ERROR ON WRITE (BIT 23)

                CRD=3E                   ; SINGLE BIT ERROR CORRECTED (BIT 30)
                RDS=3F                   ; UNCORRECTABLE DATA ERROR (BIT 31)


        ;CSR 1 CONTROL BITS

                ECC.DIS=39               ; CSR1 ECC DISABLE BIT (BIT 25)
                DIAG.CHK=3A              ; CSR1 DIAG CHK BIT (BIT 26)
                MME=3B                   ; CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
                INH.CRD=3C               ; CSR1 INHIBIT REPORT CRD BIT (BIT 28)
                TB.PAR.DIAG=3D           ; CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
                                         ;  (BIT 29)


```

```
;UBE CONTROL REGISTER BITS

  ;CONTROL REG 1
        GO=20                   ; START UBE (BIT 0)
        BR4=21                  ; ISSUE BUS REQUEST 4 (BIT 1)
        BR5=22                  ; ISSUE BUS REQUEST 5 (BIT 2)
        BR6=23                  ; ISSUE BUS REQUEST 6 (BIT 3)
        BR7=24                  ; ISSUE BUS REQUEST 7 (BIT 4)
        NPR=25                  ; ISSUE NPR (BIT 5)
        INT.ODNE=26             ; INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
        RDY=27                  ; READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
        C00=28                  ; C0 BIT (BIT 8)
        C01=29                  ; C1 BIT (BIT 9)
        FUNA=2A                 ; FUNCTION BIT A (BIT 10
        FUNB=2B                 ; FUNCTION BIT B (BIT 11)
        CC+DAT=2C               ; DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
        INH.DATIP.ROL=2D        ; INHIBIT ROL ON DATIP (BIT 13)
        DATOB.ON.DATIP=2E       ; DO DATOB AFTER DATIP CYCLE (BIT 14)
        ERR=2F                  ; EXERCISOR OR UNIBUS ERROR (BIT 15)

  ;CONTROL REG 2
        EXT.MEM0=20             ; EXTENDED MEMORY BIT 0 (BIT 0)
        EXT.MEM1=21             ; EXTENDED MEMORY BIT 1 (BIT 1)
        INH.INC.ADDR&CC=22      ; INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)
        INH.SACK=23             ; INHIBIT ASSERING BUS SACK ON BUS GRANT (BIT 3)
        PWR.DWN.SEQ=24          ; ASSERT ACLO (BIT 4)
        WNG.GNT.BCK=25          ; NO BUS GRANT RECV TO HIGEST REQUEST (BIT 5)
        MAX.LATE.ERR=26         ; NPR AND BR LATENCY TIME EXCEEDED (BIT 6)
        NO.SACK.ERR=27          ; NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)
        NO.SSYN.ERR=28          ; NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)
        WRG.A.LINES=29          ; ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)
        NO.GNT+NOT.ONE.GNT=2A   ; NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)
        INTR.SSYN.ERR=2B        ; NO SSYN RECEIVED AFTER BUS INTR (BIT 11)
        ENB.PB=2C               ; ENABLE PARITY BIT B (BIT 12)
        CCOVF=2D                ; CYCLE COUNT OVERFLOW (BIT 13)
        TM.DLY=2E               ; REQUEST BUS EVERY 10 USEC (BIT 14)

;BG6 MNEMONIC
        BG6=23                  ; BIT 3 SET FOR BG 6 ADDRESS

;ERROR AND CONTROL INFORMATION

        CONTROL.STATUS=40       ; CONTROL AND STATUS WORD
        ERROR.NUMBER=41         ; ERROR NUMBER OF CURRENT TEST
        EXPECTED.DATA=42        ; EXPECTED RESULT OF CURRENT TEST
        RECEIVED.DATA=43        ; ACTUAL RESULT OF CURRENT TEST
        ADDRESS.DATA=44         ; OTHER PERTINENT DATA
        ERROR.MASK=45           ; BITS TO CHECK IN RESULT
        MODULE.NUM=46           ; STORAGE OF MODULE NUMBER (CODED)
        LOOP.CNT=47             ; STORAGE OF LOOP COUNT FOR CONSOLE LOOP
                                ; CONTROL.
        ERROR.CONTROL=48        ; STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)
         LOE=20                 ;  LOOP ON ERROR IF SET (BIT0)
         NER=21                 ;  NO ERROR REPORT IF SET (BIT1)
         BELL=22                ;  BELL ON ERROR IF SET (BIT2)
```

```
;3810            HOE=23                  ;  HALT ON ERROR IF SET (BIT3)
;3811            SER=24                  ;  ENABLE ERROR REPORT ON CRD ERRORS IF SET
;3812            APT.PRESENT=25          ;  APT PRESENT IF SET (BIT5)
;3813            LOOP.COMMAND=26         ;  LOOP COMMAND TYPED IF SET (BIT 6)
;3814            SPECIAL=27              ;  SPECIAL BIT FOR LOOP ON SIZE TESTS (BIT 7)
;3815
;3816           APT.PARAM=49             ;APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-
;3817                                    ; WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1
;3818                                    ; AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)
;3819           APT.RESERVED=4A          ;RESERVED FOR FUTURE APT USE
;3820
;3821   ;MISCELLANEOUS CONSTANTS AND STORAGE
;3822
;3823           #AAAAAAAA=4C             ; CONSTANT
;3824           ALTER.ODD=4C             ; CONSTANT
;3825           #55555555=4D             ; CONSTANT
;3826           ALTER.EVEN=4D            ; CONSTANT
;3827           #0=4E                    ; CONSTANT
;3828           ZERO=4E                  ; CONSTANT
;3829           #FFFFFFFF=4F             ; CONSTANT
;3830           ONES=4F                  ; CONSTANT
;3831           PREVIOUS.ERROR=50        ; ERROR NUMBER OF LAST ERROR
;3832
;3833           DATA.1=51                ; STORAGE FOR FPA DATA
;3834           DATA.2=52                ; STORAGE FOR FPA DATA
;3835           DATA.3=53                ; STORAGE FOR FPA DATA
;3836           FIRST.FLT=54             ; STORAGE FOR FPA DATA
;3837           SEC.FLT=55               ; STORAGE FOR FPA DATA
;3838           THIRD.FLT=56             ; STORAGE FOR FPA DATA
;3839           T57=57                   ; TEMP LOCATION 57
;3840           T58=58                   ; TEMP LOCATION 58
;3841           T59=59                   ; TEMP LOCATION 59
;3842
;3843           BEDB=5A                  ;UBE (BUS EXERCISOR) DATA BUF REG
;3844           BECC=5B                  ;UBE (BUS EXERCISOR) CYCLE COUNT REG
;3845           BEBA=5C                  ;UBE (BUS EXERCISOR) ADDR REG
;3846           BECR1=5D                 ;UBE (BUS EXERCISOR) CONTROL REG 1
;3847           CLEAR.ERR.ADDR=5E        ;UBE CLEAR ERROR REG ADDRESS
;3848           BECR2=5F                 ;UBE (BUS EXERCISOR) CONTROL REG 2
;3849
;3850           #3(H)=60                 ; CONSTANT
;3851           #3=60                    ; CONSTANT
;3852           #12(H)=61                ; CONSTANT
;3853           #33333333=62             ; CONSTANT
;3854           #0F0F0F0F=63             ; CONSTANT
;3855           #00FF00FF=64             ; CONSTANT
;3856           #162(H)=65               ; CONSTANT FOR LS TRANSFER POINTER
;3857           BR7.IDENT=66             ; BR7 IDENTIFIER (1010(B) IN BITS 5-2)
;3858           BG7=67                   ; BG7 ADDRESS (BITS 2 AND 3 SET)
;3859
;3860   ;TEMPS FOR CPU TESTS
;3861           L30=30                   ;LS 30 TEMP (RESTORED TO 10000 AFTER USE)
;3862           L31=31                   ;LS 31 TEMP (RESTORE TO 20000 AFTER USE)
;3863           L53=53                   ;LS 53 TEMP
;3864           L73=73                   ;LS 73 TEMP
```

```

;REGISTER ADDRESSES

        CONTROL.OS=78               ; HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
        SHIFT.OS(3-0)=79            ; 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
                                    ;  (LS 20-2F)
        SHIFT.OS(4-0)=7B            ; 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
                                    ;  (LS 20-3F)
        OS=7C                       ; OS REGISTER
        DEC.CON=7D                  ; DECIMAL CARRIES
        SIZE=7E                     ; SIZE REGISTER
        MDT= 7E                     ; MEMORY REFERENCE DATA SIZE
        INTERRUPT.VEC=7E            ; HARDWARE INTERRUPT VECTOR
;
;       THIS WORD IS THE HARDWARE CONDITION CODES.  THE CC'S CAN BE READ
;       OR ARE WRITTEN HERE.  NO OTHER BITS IN THE PSL ARE EFFECTED.
;
        PSL.CC=7F                   ; PSL CONDITION CODES


.TOC    "COMMON LS ASSIGNMENTS (TO REGISTERS)"
;
;UPPER HALF OF LS
;
        XGPR.OS=0F8                 ; HARDWARE INDEX TO XGPRS
        USER.INDEX(3-0)=0F9         ; 16 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
                                    ;   AREA (LS LOCATIONS 0A0 THROUGH 0AF)
        USER.INDEX(4-0)=0FB         ; 32 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
                                    ;   AREA (LS LOCATIONS 0A0 THROUGH 0BF)
        CCR=0FC                     ; CONSOLE READ REGISTER
        TIMER=0FD                   ; INTERVAL TIMER VALUE.
        ALU.CC=0FE                  ; ALU CONDITION CODES
;
;       THIS LOCATION IS A WRITE ONLY REGISTER.  THE FOLLOWING BITS IN THE
;       PSL ARE AFFECTED:
;
;       COMPATIBILITY MODE - BIT<31>
;       CURRENT MODE - BITS<25:24>
;       INTERRUPT PRIORITY LEVEL (IPL) - BITS<20:16>
;       TRACE TRAP ENABLE (REALLY T OR TP) - BIT<4>
;       NEGATIVE CONDITION CODE - BIT<3>
;       ZERO CONDITION CODE - BIT<2>
;       OVERFLOW CONDITION CODE - BIT<1>
;       CARRY CONDITION CODE - BIT<0>
;
        PSL.HW=0FF                  ; HARDWARE PSL BITS

; These addresses are only accessible with the MOV micro instruction
; starting at address 80(H).

        WRITE.CHECK=80              ; CONSTANT 00000002
        GET.STATUS=81               ; CONSTANT 00000004
        SEEK=82                     ; CONSTANT 00000006
        READ.HEADER=83              ; CONSTANT 00000008
        WRITE.DATA=84               ; CONSTANT 0000000A
```

```
          READ.DATA=85          ; CONSTANT 0000000C
          DR0.R80=86            ; R80 IN DRIVE 0 FLAG
          DR0.RL02=87           ; RL02 IN DRIVE 0 FLAG
          DR1.RL02=88           ; RL02 IN DRIVE 1 FLAG
  ;       OPEN=89
          LABEL.SAV1=8A         ; VARIABLE TO SAVE LABEL BUT INITIALLY SET FOR THE
                                ;  VALUE OF [#43524450]
          LABEL.SAV2=8B         ; VARIABLE TO SAVE LABEL BUT INITIALLY SET FOR THE
                                ;  VALUE OF [#41434B30]
          LABEL.SAV3=8C         ; VARIABLE TO SAVE LABEL BUT INITIALLY SET FOR THE
                                ;  VALUE OF [#31202020]
          #1E=08D               ; NUMBER OF SECTORS ON A R80 TRACK IN HEX VALUE
          LABEL.READ1=8E        ; 1ST 4_CHAR. OF THE LABEL RECEIVED FROM DISK
          LABEL.READ2=8F        ; 2ND 4_CHAR. OF THE LABEL RECEIVED FROM DISK
          LABEL.READ3=90        ; 3RD 4_CHAR. OF THE LABEL RECEIVED FROM DISK
          #43524450=91          ; ASCII "CRDP"  < THESE ASCII CHARS. ARE FOR   >
          #41434B20=92          ; ASCII "ACK "  <  THE RL02 CRD PACK LABEL     >
                                ; ASCII "    "  <  CALLED "CRDPACK     "       >
  ;       OPEN=93
          #53435241=94          ; ASCII "SCRA"  < THESE ASCII CHARS. ARE FOR   >
          #54434820=95          ; ASCII "TCH "  <  THE RL02 SCRATCH PACK LABEL   >
          #20202020=96          ; ASCII "    "  <  CALLED "SCRATCH     "       >
          #1F=097               ; R80 SECTOR COUNT BIT MASK
          #1A=98                ; CONSTANT
          PORT0=99              ; RESERVED FOR PORT DATA
          PORT1=9A              ; RESERVED FOR PORT DATA
          PORT2=9B              ; RESERVED FOR PORT DATA
          PORT3=9C              ; RESERVED FOR PORT DATA
          PORT4=9D              ; RESERVED FOR PORT DATA
          PORT5=9E              ; RESERVED FOR PORT DATA
          PORT6=9F              ; RESERVED FOR PORT DATA
          PORT7=0A0             ; RESERVED FOR PORT DATA
          PORT8=0A1             ; RESERVED FOR PORT DATA
          PORT9=0A2             ; RESERVED FOR PORT DATA
          PORT10=0A3            ; RESERVED FOR PORT DATA
          PORT11=0A4            ; RESERVED FOR PORT DATA
          PORT12=0A5            ; RESERVED FOR PORT DATA
          PORT13=0A6            ; RESERVED FOR PORT DATA
          PORT14=0A7            ; RESERVED FOR PORT DATA
          PORT15=0A8            ; RESERVED FOR PORT DATA
          PORT16=0A9            ; RESERVED FOR PORT DATA
          PORT17=0AA            ; RESERVED FOR PORT DATA
          PORT18=0AB            ; RESERVED FOR PORT DATA
          PORT19=0AC            ; RESERVED FOR PORT DATA
          PORT20=0AD            ; RESERVED FOR PORT DATA
          PORT21=0AE            ; RESERVED FOR PORT DATA
          PORT22=0AF            ; RESERVED FOR PORT DATA
          PORT23=0B0            ; RESERVED FOR PORT DATA
          PORT24=0B1            ; RESERVED FOR PORT DATA
          PORT25=0B2            ; RESERVED FOR PORT DATA
          PORT26=0B3            ; RESERVED FOR PORT DATA
          PORT27=0B4            ; RESERVED FOR PORT DATA
          PORT28=0B5            ; RESERVED FOR PORT DATA
          PORT29=0B6            ; RESERVED FOR PORT DATA
          PORT30=0B7            ; RESERVED FOR PORT DATA
```

```
;3975            PORT31=0B8              ; RESERVED FOR PORT DATA
;3976   ;         OPEN=0B9
;3977   ;         OPEN=0BA
;3978   ;         OPEN=0BB
;3979   ;         OPEN=0BC
;3980   ;         OPEN=0BD
;3981   ;         OPEN=0BE
;3982   ;         OPEN=0BF
;3983            R80.SECTOR=0C0          ; STORAGE FOR R80 SECTOR SELECTION OF DAR
;3984            SECNT.TIME=0C1          ; TIME CONSTANT WAITING FOR R80 SECTOR COUNT
;3985            SEEK.CYL=0C2            ; RL02 DISK CYLINDER TO BE SOUGHT
;3986            DRIVE.STATUS=0C3        ; LOCATION CONTAINING DRIVE INFORMATION
;3987            CYL.CNTR=0C4            ; CYLINDER COUNTER
;3988            DAR.SEEK=0C5            ; DAR FORMAT FOR DOING A SEEK FUNCTION
;3989            TIMEOUT.CNTR=0C6        ; TIMEOUT COUNTER WAITING FOR CRDY AFTER A DRIVE COMMAND
;3990            WAIT.CNTR=0C7           ; LOOP COUNTER FOR WAIT TIME
;3991            WAIT.TIME1=0C8          ; GENERAL PURPOSE CONSTANT (64E00 HEX) USED TO
;3992                                    ;  TIMEOUT TESTS INSTEAD OF LEAVING IT TO MICMON
;3993            #5=0C9                  ; CONSTANT OF 5
;3994            #6=0CA                  ; CONSTANT OF 6
;3995            #7=0CB                  ; CONSTANT OF 7
;3996            #26=0CC                 ; CONSTANT OF 26(HEX)
;3997            #FFFFFFF8=0CD           ; CONSTANT
;3998            #FFFFFF80=0CE           ; CONSTANT
;3999            #FBFFFFFE=0CF           ; CONSTANT
;4000            #FFFF20C7=0D0           ; MASK FOR RL02 STATUS
;4001            #FFFF007F=0D1           ; CONSTANT
;4002            #FFFF0080=0D2           ; CONSTANT
;4003            #0000FFFF=0D3           ; CONSTANT
;4004            #00000018=0D4           ; CONSTANT
;4005            #04000001=0D5           ; CONSTANT
;4006            #00046000=0D6           ; CONSTANT FOR R80 MAX. CYL. ADDRESS
;4007            #F=0D7
;4008            #FFFFFFE0=0D8           ; MASK FOR R80 SECTOR COUNT
;4009            #FFFFFFC0=0D9           ; MASK FOR R80 STATUS
;4010            #FFFFFC00=0DA           ; MASK FOR R80 CYLINDER
;4011            #230=0DB                ; MAXIMUM CYLINDER FOR R80
;4012            #FFFFFFFE=0DC           ; INITIAL MASK FOR R80 SECTOR COUNT BIT
;4013            #FFFFFFEF=0DD           ; FINAL MASK FOR R80 SECTOR COUNT BIT
;4014            #FFFFC0FF=0DE           ; R80 STATUS ERROR MASK
;4015            #1A00=0DF               ; EXPECTED R80 STATUS
;4016            #C00=0E0                ; MASK FOR OPI & DCK BITS OF CSR STATUS
;4017            #C000=0E1               ; MASK FOR UF & MF FLAG OF R80 HEADER WORD
;4018            #B=0E2                  ; RL02 DAR BIT FORMAT FOR GET STATUS
;4019            #1D=0E3                 ; RL02 WORKING STATUS
;4020            #FFFF00FF=0E4           ; MASK FOR APT HARDWARE BYTE
;4021   ;         OPEN=0E5
;4022   ;         OPEN=0E6
;4023   ;         OPEN=0E7
;4024   ;         OPEN=0E8
;4025   ;         OPEN=0E9
;4026   ;         OPEN=0EA
;4027   ;         OPEN=0EB
;4028   ;         OPEN=0EC
;4029   ;         OPEN=0ED
```

F3

```
                ;4030  ;        OPEN=0EE
                ;4031  ;        OPEN=0EF
                ;4032  ;        OPEN=0F0
                ;4033  ;        OPEN=0F1
                ;4034  ;        OPEN=0F2
                ;4035  ;        OPEN=0F3
                ;4036  ;        OPEN=0F4
                ;4037           #D8=0F5                 ; CONSTANT
                ;4038           #0000FF80=0F6           ; CONSTANT
```

```
;4039   .page "Microinstruction Formats"
;4040   ;
;4041   ;       The following is taken directly from the NEBULA Hardware
;4042   ;       Specification.
;4043   ;
;4044   ;                          MICROINSTRUCTION FORMATS
;4045   ;
;4046   ;
;4047   ;
;4048   ;
;4049   ;                  22 21       16 15          9 8 7 6 5 4     0
;4050   ;                  +-----+-------+-------+-------+-------+-------+
;4051   ;1. BASIC          )1)     DP    )    D ADRS   ) B )CC )  SCTL   )
;4052   ;                  +-+-----------+-------------+---+---+---------+
;4053   ;
;4054   ;                  22  20 19 17 16            9 8 7 6 5 4     0
;4055   ;                  +-----+-------+-------+-------+-------+-------+
;4056   ;2. MOVE           )0 1 1) MDP )    XD ADRS    ) B )CC )  SCTL   )
;4057   ;                  +-----+-----+---------------+---+---+---------+
;4058   ;
;4059   ;                  22  20 19       14 13 11   9 8 7 6 5 4     0
;4060   ;                  +-----+-------+-------+-------+-------+-------+
;4061   ;3. EXTENDED       )0 1 0)    XDP    ) SI  ) A ) B )CC )  SCTL   )
;4062   ;                  +-----+-----------+-----+---+---+---+---------+
;4063   ;
;4064   ;                  22    19 18 16 15          9 8 7 6 5 4     0
;4065   ;                  +-----+-------+-------+-------+-------+-------+
;4066   ;4. MEM.REQ        )0 0 1 1) MF1 )   D ADRS    )MF2)DT )  SCTL   )
;4067   ;                  +-------+-----+-------------+---+---+---------+
;4068   ;
;4069   ;                  22    19 18 16 15   12 11  9 8 7 6 5 4     0
;4070   ;                  +-----+-------+-------+-------+-------+-------+
;4071   ;5. MISC           )0 0 1 0)P)0 0)   M1  ) M2  )0)R)0 0)  SCTL   )
;4072   ;                  +-------+-+---+-------+-----+-+-+---+---------+
;4073   ;
;4074   ;                  22    19 18 17                       4 3     0
;4075   ;                  +-----+-------+-------+-------+-------+-------+
;4076   ;6. JUMP           )0 0 0 1)OS)    JUMP ADDRESS          ) JCTL  )
;4077   ;                  +-------+--+--------------------------+-------+
;4078   ;
;4079   ;                  22    19 18 17      12 11  9 8 7 6 5 4 3     0
;4080   ;                  +-----+-------+-------+-------+-------+-------+
;4081   ;7. DECODE         )0 0 0 0) ) DEC.ADRS  )IFUNC) ) ) ) ) ) JCTL  )
;4082   ;                  +-------+++-----------+-----+++-+++++++-------+
;4083   ;                           )                   ) ) ) ) )
;4084   ;                  BACK UP PC          IB REQ---+ ) ) ) +--OPC/SPEC
;4085   ;                                                 ) ) )
;4086   ;                             SEL CM HI IR BYTE---+ ) +--LOAD RDEST
;4087   ;                                                   )
;4088   ;                                          LOAD OS--+
.page
;
;
;1. "BASIC" Microinstruction
;
;       CSR<22> = 1     (OPCODE)
;
;               This opcode bit causes LS Adrs <7> to be cleared.
;
;       CSR<21:16>  (Data Path Control)
;
;               This field controls the data path. It controls the 2901 ALU,
;               the ALU data source, the data destination in the 2901 array,
;               and the write to the Local Store.
;
;       CSR<15:9>   (D Adrs <6:0>)
;
;               This field selects which of the 128 accessable Local Store
;               locations to use.
;
;       CSR<8:7>    (B Adrs)
;
;               This field selects one of the four Working Registers.
;
;       CSR<6:5>    (Condition Codes)
;
;               This field specifies the data type and condition code control.
;
;       CSR<4:0>    (SCTL)
;
;               This field specifies the skip condition/micro sequencer
;               function.
```

```
.page
;
;
;2. "MOVE" Microinstruction
;
;       CSR<22:20> = 011 (OPCODE)
;
;           This combination of opcode bits allows CSR<16> to control
;           LS Adrs <7>. This permits access of LS locations 80 - FF.
;
;       CSR<19:17> (Data Path Control)
;
;           This field is decoded to produce some data path control
;           information.
;
;             0 LS[XD] <- WR[B], WR[B] <- MEM DATA* 4 LS[XD] <- MEM DATA*
;             1 MEMORY <- LS [XD]*                  5 LS[XD] <- ACC/PORT
;             2 Q <- XWR[B]                         6 LS[XD] <- WR[B] + OS
;             3 WR[B] <- LS [XD]                    7 LS[XD] <- WR[B]
;
;       CSR<16:9>  (XD<7:0>)
;
;           This field specifies which of the 256 Local Store locations to
;           use.
;
;       CSR<8:7>   (B Adrs)
;
;           This field specifies which of the four Working Registers to
;           use. For MDP = 2, this field selects either the Backup PC or
;           the VAR.
;
;       CSR<6:5>   (CC)
;
;           This field specifies the data type and condition code control.
;
;       CSR<4:0>   (SCTL)
;
;           This field specifies the Skip control. See Table 3.
;
;
;* For information on which Working Registers and Local Store locations
;  may  be  used for Memory Addresses and data source/destinations, see
;  the Memory Management documentation.  Note that  these  restrictions
;  are due to microcoding conventions and not hardware.
```

```
.page
;
;
;3. "EXTENDED" Microinstruction
;
;       CSR<22:20> = 010    (OPCODE)
;
;               This opcode causes a function internal to the 2901 array; that
;               is, an operation involving only Working Registers and the Q-
;               Register.
;
;       CSR<19:14>  (Data Path Control)
;
;               This field is decoded to supply the 2901 ALU function code,
;               the ALU Source control, the destination code, and the ALU
;               Carry-In.
;
;       CSR<13:11>  (Shift Input)
;
;               This field selects the data to be shifted into a register,
;               when a Shift function is being done.
;
;       CSR<10:9>    (A Adrs)
;
;               This field selects which Working Register to use as a source
;               of data, when RAM A is selected to the ALU.
;
;       CSR<8:7>     (B Adrs)
;
;               This field selects which Working Register to use as a source
;               (when RAM B is selected to the ALU) and/or destination (when
;               the RAM is written) of data.
;
;       CSR<6:5>     (CC)
;
;               This field specifies the data type and condition code control.
;
;       CSR<4:0>     (SCTL)
;
;               This field specifies the skip condition/micro sequencer
;               function.
```

```
.page
;
;
;4. "MEM.REQ" Microinstruction
;
;       CSR<22:19> = 0011     (OPCODE)
;
;           This opcode causes a Memory Request to be started. The Local
;           Store is output on on the D-Bus, to be used as the virtual
;           address. Working Register 5 is written with this address.
;
;       CSR<18:16>, <8:7>    (Memory Function)
;
;           This field specifies to the Memory Controller the type of
;           request: physical, virtual, type of access, etc. For the
;           list of functions and the codes, see the NEBULA Memery Spec.
;
;       CSR<15:9>    (D Adrs <6:0>)
;
;           This field selects which of the 128 accessable Local Store
;           locations to use as the virtual address.
;
;       CSR<6:5>     (Data Type)
;
;           This field specifies the data type of the request.
;
;                   00:=BYTE
;                   01:=WORD
;                   10:=SIZE Reg
;                   11:=LONG
;
;       CSR<4:0>     (SCTL)
;
;           This field specifies the skip condition/micro sequencer
;           function.
```

```
                ;4241   .page
                ;4242   ;
                ;4243   ;
                ;4244   ;5. "MISC" Microinstruction
                ;4245   ;
                ;4246   ;       CSR<22:19> = 0010 (OPCODE)
                ;4247   ;
                ;4248   ;           This combination of opcode bits causes the data path to no-op,
                ;4249   ;           and enables the MISC decoders.
                ;4250   ;
                ;4251   ;       CSR<18>                 (Port/Misc Select)
                ;4252   ;
                ;4253   ;           This bit defines the instruction to be either a Misc or a Port
                ;4254   ;           instruction.   For CSR<18> = 1, CSR<17:10> is the Command Byte
                ;4255   ;           to the Port, CSR<9:5> must be zero and CSR<4:0>  is  the  SCTL
                ;4256   ;           field.  For CSR<18> = 0, it is a Misc instruction and the rest
                ;4257   ;           of the word is defined as follows.
                ;4258   ;
                ;4259   ;       CSR<17:16>              (Not Used)
                ;4260   ;
                ;4261   ;       CSR<15:12>  (Misc Function 1)
                ;4262   ;
                ;4263   ;           These four bits are decoded to do the following functions:
                ;4264   ;
                ;4265   ;                   0 = CLEAR State Reg<1:0>
                ;4266   ;                   1 = CLEAR State Reg<1>, SET State Reg<0>
                ;4267   ;                   2 = SET State Reg<1>, CLEAR State Reg<0>
                ;4268   ;                   3 = CLEAR State Reg<0>
                ;4269   ;                   4 = CLEAR State Reg<1>
                ;4270   ;                   5 = SET State Reg<0>
                ;4271   ;                   6 = SET State Reg<1>
                ;4272   ;                   7 = SET REGISTER BACKUP MASK FLAG
                ;4273   ;                   8 = SET ACC I/O MODE
                ;4274   ;                   9 = SET PORT I/O MODE
                ;4275   ;                   A = CLEAR WCS HI ADRS
                ;4276   ;                   B = SET WCS HI ADRS
                ;4277   ;                   C = CLEAR CPU ATTN AND CPU ACK
                ;4278   ;                   D = SET CPU ACK
                ;4279   ;                   E = SET CPU ATTN
                ;4280   ;                   F = NOP
                ;4281   ;
                ;4282   ;       CSR<11:9>   (Misc Function 2)
                ;4283   ;
                ;4284   ;           This field is decoded to do the following functions:
                ;4285   ;
                ;4286   ;                   0 = NOP
                ;4287   ;                   1 = Mask IRD Interrupt Detection during Next State
                ;4288   ;                   2 = Assert CPU Data Available and pass WR to ACC/Port
                ;4289   ;                   3 = Trap ACC
                ;4290   ;                   4 = Read ACC uPC
                ;4291   ;                   5 = Assert XFER GRANT to Port
                ;4292   ;                   6 = Mask Halt and T-Bit Traps
                ;4293   ;                   7 = Write WR to Console Read Register
```

```
;4294   .page
;4295   ;
;4296   ;       CSR<8>          (MUST BE ZERO)
;4297   ;
;4298   ;       CSR<7>          (WR Address)
;4299   ;
;4300   ;
;4301   ;           This bit selects WR[0] or WR[1] to be put on the Y-Bus.
;4302   ;
;4303   ;       CSR<6:5>                (Not Used)
;4304   ;
;4305   ;       CSR<4:0>        (SCTL)
;4306   ;
;4307   ;           This  field  specifies  the  skip  condition/micro   sequencer
;4308   ;           function.
;4309   ;
```

```
;4310  .page
;4311  ;
;4312  ;
;4313  ;6. "JUMP" Microinstruction
;4314  ;
;4315  ;     CSR<22:19> = 0001    (OPCODE)
;4316  ;
;4317  ;          This opcode causes the data path to no-op, and the micro
;4318  ;          sequencer to execute a JUMP.
;4319  ;
;4320  ;     CSR<18>       (Select OS)
;4321  ;
;4322  ;          This bit, when asserted, causes OS<4:0> to be OR'ed with the
;4323  ;          five low bits of the of the jump microaddress.
;4324  ;
;4325  ;     CSR<17:4>     (Jump Microaddress)
;4326  ;
;4327  ;          If Select OS is CLEAR, this field specifies the fourteen-bit
;4328  ;          jump micro address. If Select OS is SET, CSR<17:9> is Jump
;4329  ;          Adrs<13:5> and OS<4:0> OR'ed with CSR<8:4> is Jump Adrs<4:0>.
;4330  ;
;4331  ;     CSR<3:0>      (JCTL)
;4332  ;
;4333  ;          This field specifies the jump condition/micro sequencer
;4334  ;          function.
```

```
.page
;
;
;7. "DECODE" Microinstruction
;
;    A.  DESCRIPTION
;
;    CSR<22:19> = 0000 (OPCODE)
;
;        This combination of opcode bits  causes  the  Local  Store  to
;        access location 10 (the PC.) The Local Store drives the D-Bus.
;        The 2901 is set up to input the data on the  D-Bus,  increment
;        it,  and output to the Y-Bus.  Thus, the Y-Bus contains the PC
;        + 1.
;
;    CSR<18>      (BPC [asserted low])
;
;        If this bit is CLEAR, the incremented PC is written  into  the
;        2901 RAM.  If it is SET, the RAM is not written.
;
;    CSR<17:12>   (DEC.ADRS <13:8>)
;
;        This data is taken to be  the  upper  six  bits  of  the  next
;        microaddress.  The low eight bits are supplied by the IRD ROM.
;        The actual JUMP/JSR/NO-OP is determined by the JCTL field.
;
;    CSR<11:9>    (IFUNC)
;
;        This is the IFUNC field.   It  defines  which  fork  is  being
;        taken.
;
;    CSR<8>       (IB.REQ)
;
;        This is the IB.REQ bit.  When it  is  CLEAR,  the  LS  is  not
;        written,  no  interaction  with  memory  is  initiated and the
;        instructin buffer is not affected.
;
;        When it is SET:
;
;        The incremented PC (valid on the  Y-Bus  by  the  end  of  the
;        cycle)   is  re-written  to  Local  Store  location  0.   This
;        completes the PC increment function.
;
;        A Conditional Memory Request is executed.  If the two low bits
;        of  the  PC  are not 11, the request is not initiated.  If the
;        two  low  bits  are  set,  an  IB  PREFETCH  is  done.     The
;        (unincremented)  PC  is output to the memory and the CPU grant
;        (CPUG) signal is examined.  If CPUG is low by  T120,  the  CPU
;        will  stall  until  the  CPUG  signal becomes true.  After the
;        request is completed, the next state entered is dependent upon
;        the JCTL field.
```

```
.page
;
;       CSR<7>      (SEL CM HI IR BYTE)
;
;           This bit enables the upper  byte  of  the  Compatibility  Mode
;           instruction  in the Prefetch register to drive the IB-Bus.  It
;           is used only while in Compatibility Mode, when  IRD  is  being
;           done.  It must never be asserted in VAX Mode.
;
;
;       CSR<6>       (LD.OS)
;
;           If this bit is SET, the eight-bit data  on  the  IB  Bus  (the
;           output  of  the  instruction  buffer)  is  loaded  into the OS
;           register.  This load is dependent upon Compatibility Mode  and
;           LD  R  Dest  (If  CM.AND.LD R Dest, OS<3> <- 0.) If the bit is
;           CLEAR, OS is not affected.
;
;       CSR<5>       (LD R DEST)
;
;           If this bit is SET, the R  Dest  Skip  bit  will  be  SET  for
;           Specifier Mode equal to 5 and CLEARED for Mode not equal to 5,
;           in VAX mode, or mode 0 while in Compatibility Mode.   If  this
;           bit is CLEAR, the R Dest bit will be unaffected.
;
;       CSR<4>              (OPC/SPEC)
;
;           This bit (effectively another IFUNC bit) defines the  data  to
;           be  decoded;   that is, if it is an Opcode or a Specifier.  In
;           hardware, it selects which ROM is to drive the  micro  address
;           bus.
;
;       CSR<3:0>     (JCTL)
;
;           These four bits are decoded to make one of 16 functions.   See
;           Table 3.
```

```
.page "Tables"
;
;                                   TABLES
;
;
;
;                   Table 1.  2901 Control Decoding
;
;
;          The 2901 control decoder examines CSR<22:14>. This nine-bit
;          field  of  the microword changes in meaning for the different
;          cases of micro opcode.
;
;          CSR      22  21  20  19  18  17  16  15  14
;                   --  --  --  --  --  --  --  --  --
;1. BASIC            1  [Six bit function code]  X   X
;2. MOVE             0   1   1  [function]   X   X   X
;3. EXTENDED         0   1   0  [Six bit function code]
;4. MEM.REQ          0   0   1   1   X   X   X   X   X      WR[5] <- D
;5. MISC             0   0   1   0   X   X   X   X   X      Y-Bus <- WR
;6. JUMP             0   0   0   1   X   X   X   X   X      NO-OP
;7. DECODE           0   0   0   0  BPC* X   X   X   X
;
;  If BPC = 0, Write WR; else, No-op
;
;       This decoder is implemented using a ROM and a PAL. 512
;       locations are needed for CSR<22:14> to index into. The
;       decoding logic supplies 10 control bits:
;
;                   ALU Source Select   (3)
;                   ALU Function   (3)
;                   ALU Result Destination   (3)
;                   ALU Carry-In   (1)
;
;       The ROM addresses are therefore allocated in the following way:
;
;       Location      0 -   F        Incr. D-Bus, Output ALU, No Write.
;                    10 -  1F        Incr. D-Bus, Output ALU, Write RAM.
;                    20 -  3F        No-Op.
;                    40 -  5F        Output WR, bypassing ALU.
;                    60 -  7F        Pass D-Bus through ALU and write RAM.
;                    80 -  BF        CSR<19:14> indicates 2901 function.
;                    C0 -  FF        CSR<19:17> indicates 2901 function.
;                   100 - 1FF        CSR<21:16> indicates 2901 function.
;
;       For the detailed function table see the Microcode Listing.
```

```
.page
;
;
;                   Table 2. Local Store Write
;
;     The Local Store Write Controller examines CSR<22:17>, CSR<6>, the
;     SIZE  register  (SIZE<1:0>)  and the Compatibility Mode bit (CM).
;     These eight bits are decoded to determine which parts of  the  LS
;     to  write, if any.  The LS is written with data from the 2901 ALU
;     (Y-Bus).
;
;                  CSR      22  21  20  19  18  17
;                           --  --  --  --  --  --
;        1. BASIC            1   0   X   X   X   X    No Write
;                            1   1   X   X   X   X    Write *
;        2. MOVE             0   1   1   0   X   1    No Write
;                            0   1   1   0   1   X    No Write
;                            0   1   1   1   X   X    Write *
;        3. EXTENDED         0   1   1   X   X   X    No Write
;        4. MEM.REQ          0   0   1   1   X   X    No Write
;        5. MISC             0   0   1   0   X   X    No Write
;        6. JUMP             0   0   0   1   X   X    No Write
;        7. DECODE           0   0   0   0   X   X    Conditional Write **
;
; * LS Write, Data Type Dependent. These cases now examine CSR<6>.
;
;    If CSR<6> = 0, and CM = 0   Write LS<31:0>  (Data Type = LONG)
;    If CSR<6> = 0, and CM = 1   Write LS<15:0>  (Compatibility
;                                                 Mode. DT = WORD)
;
;    If CSR<6> = 1, the Write is conditional on the SIZE register.
;
;    If CM = 0   SIZE<1:0> = 00  Write LS<7:0>  (BYTE)
;                          = 01  Write LS<15:0> (WORD)
;                          = 1X  Write LS<31:0> (LONG)
;
;    If CM = 1  Write LS<15:0>  (WORD)
;
; ** If CSR<8> = 0, No Write
;     If CSR<8> = 1, Write LS<31:0>  (Writes incremented PC)
```

```
.page
;
;
;                    Table 3. SKIP/JUMP Conditions
;
;This table summarizes the Skip and Jump conditions, and the other
;special functions of the micro sequencer.
;
;SCTL codes 0-F, 17-1F are Skip conditions. The codes 10-17 execute
;special functions. JCTL codes 0-B are Jump conditions, and C-F
;execute special functions.
;
;
;Sctl    Function                 Sctl    Function
;----    --------                 ----    --------
; 00     Not ALU N                 10     RTN+1 if No Mem Err
; 01     Not ALU Z                 11     Loop if Not ALU Z
; 02     Not ALU V                 12     POP Stack
; 03     ALU C                     13     Loop if ALU C
; 04     Not PSL C                 14     Return
; 05     Branch False              15     No Skip (NOP)
; 06     RDEST                     16     RTN+1
; 07     Not Inter. Req.           17     Loop if No Inter. and No Sync
; 08     ALU N                     18     Console ATTN
; 09     ALU Z                     19     Console ACK
; 0A     ALU V                     1A     Port Interrupt Pending
; 0B     Not ALU C                 1B     Reg. Backup Mask Flag
; 0C     State 0                   1C     ALU N.XOR.ALU V
; 0D     State 1                   1D     Not Memory Error
; 0E     Not State 0               1E     Skip
; 0F     Not State 1               1F     Sync
;
;
;Jctl    Function                 Jctl    Function
;----    --------                 ----    --------
; 00     Not ALU N                 08     ALU N
; 01     Not ALU Z                 09     ALU Z
; 02     Not ALU V                 0A     ALU V
; 03     ALU C                     0B     Not ALU C
; 04     JUMP                      0C     JSR
; 05     Branch False              0D     JSR if IB Valid
; 06     RDEST                     0E     No Jump; Skip if IB Valid
; 07     Not Interr. Req           0F     IB Valid
;

```

```
.PAGE  "2901 ALU Control Description"

;
;                The 2901 ALU is the heart  of  the  CPU  module.   It  is
;           responsible for all CPU calculations.  The control lines going
;           to I0 through I8 control the function, input,  and  output  of
;           the  2901.   When  a  failure  involving  the 2901 occurs, the
;           logical place to start looking  is  the  control  lines.   The
;           following  tables  describe what the control signals should be
;           for various functions.  Use it whenever the control lines  are
;           to  be  checked.  The test descriptions tell the operator what
;           function the 2901 is supposed to be doing.
;
;                The SRC CTL on lines I0 through I2 (pins  12  through  14
;           respectively)  control  where  the  inputs to the internal ALU
;           within the 2901 come from.  The inputs to the internal ALU are
;           labeled R and S.  R is one operand (4 bits) and S is the other
;           operand (also 4 bits).  The R input can come  from  a  Working
;           Register  (labeled  A), the D bus (direct input), or be forced
;           to 0.  The S  input  can  come  from  either  set  of  working
;           registers  (labeled  A and B), the Q register, or forced to 0.
;           A table of these control signals follow:
;
;
;            ----------------------------
;           )              )    )  SRC    )
;           )              )    )operands)
;           )              )octal)--------)
;           ) I2  I1  I0   )code )  R  S  )
;           )----------------------------)
;           ) L   L   L    )  0  )  A  Q  )
;           ) L   L   H    )  1  )  A  B  )
;           ) L   H   L    )  2  )  0  Q  )
;           ) L   H   H    )  3  )  0  B  )
;           ) H   L   L    )  4  )  0  A  )
;           ) H   L   H    )  5  )  D  A  )
;           ) H   H   L    )  6  )  D  Q  )
;           ) H   H   H    )  7  )  D  0  )
;            ----------------------------
;
;             ALU SOURCE OPERAND CONTROL
;
;
;
;
;                The ALU CTL on lines I3  through  I5  (pins  26,  28,  27
;           respectively)  control  what function the ALU is to perform on
;           the two operands R and S.  A table follows:
```

```
.page
;
;
;               --------------------------------------
;               )                )octal)    ALU      )
;               ) I5   I4   I3   )code )  function   )
;               )-------------------------------------)
;               ) L    L    L    )  0  )R PLUS S     )
;               ) L    L    H    )  1  )S MINUS R    )
;               ) L    H    L    )  2  )R MINUS S    )
;               ) L    H    H    )  3  )R OR S       )
;               ) H    L    L    )  4  )R AND S      )
;               ) H    L    H    )  5  )R NOT AND S)
;               ) H    H    L    )  6  )R XOR S      )
;               ) H    H    H    )  7  )R XNOR S     )
;               --------------------------------------
;                    ALU FUNCTION CONTROL
;
;
;
;
;                    The DST CTL  on  lines  I6  through  I8  (pins  5,  7,  6
;               respectively) control what goes out onto the Y bus (either the
;               internal ALU   result   or  a  WR  directly)  and  what  happens
;               internal  to  the 2901 (shifting, writing the Q register etc).
;               Internal shifting left or  right  is  accomplished  with  this
;               control.   Below is a table explaining the various destination
;               possibilities.  The arrow refers to where the output will  end
;               up.   B  indicates the WR addressed by the B ADDR, F refers to
;               the output of the internal ALU, and A refers to the output  of
;               the  WR  addressed by the A ADDR.  Up is toward the MSB, while
;               down is toward the LSB.
;
;
;
;  ---------------------------------------------------------------------------------
;  )nomenclature)                )octal) RAM function    ) Q-reg function)    Y    )
;  ) in listing ) I8   I7   I6   )code ) shift ) load    ) shift ) load   ) output)
;  )-------------------------------------------------------------------------------)
;  ) LOAD Q       )  L    L    L )  0  ) NONE  ) NONE    ) NONE  ) F->Q   )    F    )
;  )              )              )     )       )         )       )        )         )
;  ) NOP          )  L    L    H )  1  ) NONE  ) NONE    ) NONE  ) NONE   )    F    )
;  )              )              )     )       )         )       )        )         )
;  ) WRITE.B.A    )  L    H    L )  2  ) NONE  ) F->B    ) NONE  ) NONE   )    A    )
;  )              )              )     )       )         )       )        )         )
;  ) WRITE.B.F    )  L    H    H )  3  ) NONE  ) F->B    ) NONE  ) NONE   )    F    )
;  )              )              )     )       )         )       )        )         )
;  ) RSHF.RAM.Q   )  H    L    L )  4  ) DOWN  ) F/2->B) DOWN  ) Q/2->Q)    F    )
;  )              )              )     )       )         )       )        )         )
;  ) RSHF.RAM     )  H    L    H )  5  ) DOWN  ) F/2->B) NONE  ) NONE   )    F    )
;  )              )              )     )       )         )       )        )         )
;  ) LSHF.RAM.Q   )  H    H    L )  6  )  UP   ) 2F->B   )  UP   ) 2Q->Q  )    F    )
;  )              )              )     )       )         )       )        )         )
;  ) LSHF.RAM     )  H    H    H )  7  )  UP   ) 2F->B   ) NONE  ) NONE   )    F    )
;  ---------------------------------------------------------------------------------
;                           ALU DESTINATION CONTROL
;
```

```
                ;4655   .page
```

```
;4656  .PAGE
;4657  .TOC     "DIAGNOSTIC MACROS"
;4658
;4659  LS.DATA.WORD/=<15:0>
;4660  UPPER.BYTE/=<23:16>
;4661
;4662  WORD[]             "UPPER.BYTE/0,LS.DATA.WORD/@1"
;4663  COUNT.LS[]         "UPPER.BYTE/0,LS.DATA.WORD/@1"
;4664  COUNT.MM[]         "UPPER.BYTE/1,LS.DATA.WORD/@1"
;4665  START.ADR[]        "UPPER.BYTE/0,LS.DATA.WORD/@1"
;4666
;4667
;4668
;4669  ASCII.LIT/=<15:0>
;4670
;4671           CA=4341          ;ASCII VALUE FOR FILE NAMES
;4672           CB=4342          ;  !
;4673           CC=4343          ;  !
;4674           CD=4344          ;  !
;4675           CE=4345          ;  V
;4676           CF=4346
;4677           CG=4347
;4678           CH=4348
;4679
;4680
;4681  ASCII []           "UPPER.BYTE/0,ASCII.LIT/@1"
;4682
;4683
;4684  .REGION/0,6F
```

```
                     ;4685  .PAGE    "TEST POINTERS"
                     ;4686  ;
                     ;4687  ;        This portion contains the pointers to all tests in this section.  The
                     ;4688  ;        WCS address of the JUMP instruction corresponds to the test number.
                     ;4689  ;        E.g. WCS 5 contains a JUMP to test 5.  All unused test numbers will
                     ;4690  ;        contain a JUMP to an error routine.  This portion is 80. locations
                     ;4691  ;        long, allowing for up to 80. tests per section, from WCS address 1 to
                     ;4692  ;        WCS address 4F.
                     ;4693  ;
                     ;4694  1:                                      ;TEST POINTERS BEGIN AT WCS ADDRESS 1
                     ;4695
                     ;4696
                     ;4697
U 0001, 88,7624      ;4698           JMP [T.1]                      ;GO TO TEST 1
U 0002, 88,7914      ;4699           JMP [T.2]                      ;GO TO TEST 2
U 0003, 88,7C24      ;4700           JMP [T.3]                      ;GO TO TEST 3
U 0004, 88,81C4      ;4701           JMP [T.4]                      ;GO TO TEST 4
U 0005, 08,8564      ;4702           JMP [T.5]                      ;GO TO TEST 5
U 0006, 08,8B74      ;4703           JMP [T.6]                      ;GO TO TEST 6
U 0007, 08,9324      ;4704           JMP [T.7]                      ;GO TO TEST 7
U 0008, 08,9A14      ;4705           JMP [T.8]                      ;GO TO TEST 8
U 0009, 08,9F84      ;4706           JMP [T.9]                      ;GO TO TEST 9
U 000A, 08,A2F4      ;4707           JMP [T.A]                      ;GO TO TEST A
U 000B, 88,A7D4      ;4708           JMP [T.B]                      ;GO TO TEST B
U 000C, 88,B384      ;4709           JMP [T.C]                      ;GO TO TEST C
U 000D, 88,B644      ;4710           JMP [T.D]                      ;GO TO TEST D
U 000E, 08,C044      ;4711           JMP [T.E]                      ;GO TO TEST E
U 000F, 08,C7F4      ;4712           JMP [T.F]                      ;GO TO TEST F
U 0010, 08,DB84      ;4713           JMP [T.10]                     ;GO TO TEST 10
U 0011, 88,DF84      ;4714           JMP [T.11]                     ;GO TO TEST 11
U 0012, 88,6BE4      ;4715           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4716                                          ; IN THIS OVERLAY
U 0013, 88,6BE4      ;4717           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4718                                          ; IN THIS OVERLAY
U 0014, 88,6BE4      ;4719           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4720                                          ; IN THIS OVERLAY
U 0015, 88,6BE4      ;4721           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4722                                          ; IN THIS OVERLAY
U 0016, 88,6BE4      ;4723           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4724                                          ; IN THIS OVERLAY
U 0017, 88,6BE4      ;4725           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4726                                          ; IN THIS OVERLAY
U 0018, 88,6BE4      ;4727           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4728                                          ; IN THIS OVERLAY
U 0019, 88,6BE4      ;4729           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4730                                          ; IN THIS OVERLAY
U 001A, 88,6BE4      ;4731           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4732                                          ; IN THIS OVERLAY
U 001B, 88,6BE4      ;4733           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4734                                          ; IN THIS OVERLAY
U 001C, 88,6BE4      ;4735           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4736                                          ; IN THIS OVERLAY
U 001D, 88,6BE4      ;4737           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                     ;4738                                          ; IN THIS OVERLAY
U 001E, 88,6BE4      ;4739           JMP [ERR.NO.TEST]              ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
```

```
                    ;4740                                          ; IN THIS OVERLAY
U 001F, 88,6BE4     ;4741          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4742                                          ; IN THIS OVERLAY
U 0020, 88,6BE4     ;4743          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4744                                          ; IN THIS OVERLAY
U 0021, 88,6BE4     ;4745          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4746                                          ; IN THIS OVERLAY
U 0022, 88,6BE4     ;4747          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4748                                          ; IN THIS OVERLAY
U 0023, 88,6BE4     ;4749          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4750                                          ; IN THIS OVERLAY
U 0024, 88,6BE4     ;4751          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4752                                          ; IN THIS OVERLAY
U 0025, 88,6BE4     ;4753          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4754                                          ; IN THIS OVERLAY
U 0026, 88,6BE4     ;4755          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4756                                          ; IN THIS OVERLAY
U 0027, 88,6BE4     ;4757          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4758                                          ; IN THIS OVERLAY
U 0028, 88,6BE4     ;4759          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4760                                          ; IN THIS OVERLAY
U 0029, 88,6BE4     ;4761          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4762                                          ; IN THIS OVERLAY
U 002A, 88,6BE4     ;4763          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4764                                          ; IN THIS OVERLAY
U 002B, 88,6BE4     ;4765          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4766                                          ; IN THIS OVERLAY
U 002C, 88,6BE4     ;4767          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4768                                          ; IN THIS OVERLAY
U 002D, 88,6BE4     ;4769          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4770                                          ; IN THIS OVERLAY
U 002E, 88,6BE4     ;4771          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4772                                          ; IN THIS OVERLAY
U 002F, 88,6BE4     ;4773          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4774                                          ; IN THIS OVERLAY
U 0030, 88,6BE4     ;4775          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4776                                          ; IN THIS OVERLAY
U 0031, 88,6BE4     ;4777          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4778                                          ; IN THIS OVERLAY
U 0032, 88,6BE4     ;4779          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4780                                          ; IN THIS OVERLAY
U 0033, 88,6BE4     ;4781          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4782                                          ; IN THIS OVERLAY
U 0034, 88,6BE4     ;4783          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4784                                          ; IN THIS OVERLAY
U 0035, 88,6BE4     ;4785          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4786                                          ; IN THIS OVERLAY
U 0036, 88,6BE4     ;4787          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4788                                          ; IN THIS OVERLAY
U 0037, 88,6BE4     ;4789          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4790                                          ; IN THIS OVERLAY
U 0038, 88,6BE4     ;4791          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4792                                          ; IN THIS OVERLAY
U 0039, 88,6BE4     ;4793          JMP [ERR.NO.TEST]               ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                    ;4794                                          ; IN THIS OVERLAY
```

```
U 003A, 88,6BE4   ;4795            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4796                                             ; IN THIS OVERLAY
U 003B, 88,6BE4   ;4797            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4798                                             ; IN THIS OVERLAY
U 003C, 88,6BE4   ;4799            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4800                                             ; IN THIS OVERLAY
U 003D, 88,6BE4   ;4801            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4802                                             ; IN THIS OVERLAY
U 003E, 88,6BE4   ;4803            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4804                                             ; IN THIS OVERLAY
U 003F, 88,6BE4   ;4805            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4806                                             ; IN THIS OVERLAY
U 0040, 88,6BE4   ;4807            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4808                                             ; IN THIS OVERLAY
U 0041, 88,6BE4   ;4809            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4810                                             ; IN THIS OVERLAY
U 0042, 88,6BE4   ;4811            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4812                                             ; IN THIS OVERLAY
U 0043, 88,6BE4   ;4813            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4814                                             ; IN THIS OVERLAY
U 0044, 88,6BE4   ;4815            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4816                                             ; IN THIS OVERLAY
U 0045, 88,6BE4   ;4817            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4818                                             ; IN THIS OVERLAY
U 0046, 88,6BE4   ;4819            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4820                                             ; IN THIS OVERLAY
U 0047, 88,6BE4   ;4821            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4822                                             ; IN THIS OVERLAY
U 0048, 88,6BE4   ;4823            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4824                                             ; IN THIS OVERLAY
U 0049, 88,6BE4   ;4825            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4826                                             ; IN THIS OVERLAY
U 004A, 88,6BE4   ;4827            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4828                                             ; IN THIS OVERLAY
U 004B, 88,6BE4   ;4829            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4830                                             ; IN THIS OVERLAY
U 004C, 88,6BE4   ;4831            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4832                                             ; IN THIS OVERLAY
U 004D, 88,6BE4   ;4833            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4834                                             ; IN THIS OVERLAY
U 004E, 88,6BE4   ;4835            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4836                                             ; IN THIS OVERLAY
U 004F, 88,6BE4   ;4837            JMP [ERR.NO.TEST]                ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                  ;4838                                             ; IN THIS OVERLAY
```

```
                   ;4839  .PAGE    "DIAGNOSTIC POINTERS"
                   ;4840  ;
                   ;4841  ;        This section contains all the pointers needed for this diagnostic
                   ;4842  ;        overlay.  The first pointer (at WCS location 0) points to a data
                   ;4843  ;        transfer routine.  It is the first location executed after a new
                   ;4844  ;        WCS load.  The insruction here is a JUMP to TRANSFER.POINT which may
                   ;4845  ;        contain instructions to transfer data.  If no data is to be trans-
                   ;4846  ;        ferred, or at the completion of the data transfers, the CPU will
                   ;4847  ;        issue CPU ATTN with all bits of the control and status word clear.
                   ;4848  ;
                   ;4849  ;        The next 80. locations (from WCS location 1 to WCS location 4F)
                   ;4850  ;        contain pointers to each test in this overlay.  The pointers are
                   ;4851  ;        JUMP instructions to the test number corresponding to the location
                   ;4852  ;        number.  E.g. location 5 will contain a JUMP to test 5.  All unused
                   ;4853  ;        test numbers will contain a JUMP to an error routine.  This portion
                   ;4854  ;        appears after the definitions in this listing.
                   ;4855  ;
                   ;4856  ;        Finally the next 32. locations (from WCS location 50 to 6F) contain
                   ;4857  ;        pointers (Jump instructions) to WCS subroutines which are to be used by
                   ;4858  ;        the console diagnostic monitor.
                   ;4859  ;
                   ;4860
U 0000, 08,6C24    ;4861  0:       JMP [TRANSFER.POINT]              ;GO TO ROUTINE THAT LOADS LS 7 WITH
                   ;4862                                             ; POINTER TO DATA SECTION AND ISSUES
                   ;4863                                             ; CPU ATTN WITH TRANSFER BIT SET.
                   ;4864
                   ;4865  50:                                        ;START AT WCS LOCATION 50 FOR SUB-
                   ;4866                                             ; ROUTINE POINTERS
                   ;4867  ;
                   ;4868  ; SUBROUTINE POINTERS
                   ;4869  ;
U 0050, 08,6084    ;4870           JMP [DEPOSIT.CSR]                 ;GO TO WCS SUBROUTINE WHICH WRITES THE
                   ;4871                                             ; MCT CSR FOR DEPOSIT CSR COMMAND
U 0051, 08,60D4    ;4872           JMP [EXAMINE.CSR]                 ;GO TO WCS SUBROUTINE WHICH READS THE
                   ;4873                                             ; MCT CSR FOR EXAMINE CSR COMMAND
U 0052, 08,61F4    ;4874           JMP [DEPOSIT.TB]                  ;GO TO WCS SUBROUTINE WHICH WRITES THE
                   ;4875                                             ; MCT TRANSLATION BUFFER FOR DEPOSIT
                   ;4876                                             ; TB COMMAND
U 0053, 88,63A4    ;4877           JMP [EXAMINE.TB]                  ;GO TO WCS SUBROUTINE WHICH READS THE
                   ;4878                                             ; MCT TRANSLATION BUFFER FOR EXAMINE
                   ;4879                                             ; TB COMMAND
U 0054, 88,65C4    ;4880           JMP [DEPOSIT.MM]                  ;GO TO WCS SUBROUTINE WHICH WRITES MAIN
                   ;4881                                             ; MEMORY FOR DEPOSIT MM COMMAND.  ALSO
                   ;4882                                             ; USED TRANSFERING DATA FROM WCS TO
                   ;4883                                             ; MAIN MEMORY.
U 0055, 08,6624    ;4884           JMP [EXAMINE.MM]                  ;GO TO WCS SUBROUTINE WHICH READS MAIN
                   ;4885                                             ; MEMORY FOR EXAMINE MM COMMAND.
U 0056, 08,6894    ;4886           JMP [SAVE.WR]                     ;GO TO WCS SUBROUTINE WHICH STORES THE
                   ;4887                                             ; CONTENTS OF WR0-WR3 IN LS 0-3
U 0057, 88,68E4    ;4888           JMP [RESTORE.WR]                  ;GO TO WCS SUBROUTINE WHICH RESTORES THE
                   ;4889                                             ; CONTENTS OF WR0-WR3 FROM LS 0-3
U 0058, 88,6424    ;4890           JMP [DEPOSIT.UBS]                 ;GO TO WCS SUBROUTINE WHICH WRITES THE
                   ;4891                                             ; UNIBUS MAP ON THE MEMORY CONTROLLER
U 0059, 08,6544    ;4892           JMP [EXAMINE.UBS]                 ;GO TO WCS SUBROUTINE WHICH READS THE
                   ;4893                                             ; UNIBUS MAP ON THE MEMORY CONTROLLER
```

```
U 005A, 08,6684  ;4894          JMP [EXAMINE.ICSR]                ;GO TO WCS SUBROUTINE WHICH READS THE
                 ;4895                                            ; IDC CSR
U 005B, 08,66B4  ;4896          JMP [EXAMINE.IDAR]                ;GO TO WCS SUBROUTINE WHICH READS THE
                 ;4897                                            ; IDC DAR
U 005C, 08,66E4  ;4898          JMP [EXAMINE.DBUF]                ;GO TO WCS SUBROUTINE WHICH READS THE
                 ;4899                                            ; IDC DBUF
U 005D, 88,6714  ;4900          JMP [EXAMINE.PATT]                ;GO TO WCS SUBROUTINE WHICH READS THE
                 ;4901                                            ; IDC ECC PATTERN
U 005E, 88,6744  ;4902          JMP [EXAMINE.POSIT]               ;GO TO WCS SUBROUTINE WHICH READS THE
                 ;4903                                            ; IDC ECC POSITION
U 005F, 08,6794  ;4904          JMP [DEPOSIT.ICSR]                ;GO TO WCS SUBROUTINE WHICH WRITES THE
                 ;4905                                            ; IDC CSR
U 0060, 08,67C4  ;4906          JMP [DEPOSIT.IDAR]                ;GO TO WCS SUBROUTINE WHICH WRITES THE
                 ;4907                                            ; IDC DAR
U 0061, 08,67F4  ;4908          JMP [DEPOSIT.DBUF]                ;GO TO WCS SUBROUTINE WHICH WRITES THE
                 ;4909                                            ; IDC DBUF
U 0062, 88,6824  ;4910          JMP [CLEAR.FIFO.ADDR]             ;GO TO WCS SUBROUTINE WHICH CLEARS
                 ;4911                                            ; THE IDC FIFO ADDRESS
U 0063, 88,6844  ;4912          JMP [SELECT.FIFO.A]               ;GO TO WCS SUBROUTINE WHICH SELECTS
                 ;4913                                            ; FIFO A
U 0064, 08,6864  ;4914          JMP [SELECT.FIFO.B]               ;GO TO WCS SUBROUTINE WHICH SELECTS
                 ;4915                                            ; FIFO B
U 0065, DB,0015  ;4916          NOP                               ;NOT USED
U 0066, DB,0015  ;4917          NOP                               ;NOT USED
U 0067, DB,0015  ;4918          NOP                               ;NOT USED
U 0068, DB,0015  ;4919          NOP                               ;NOT USED
U 0069, DB,0015  ;4920          NOP                               ;NOT USED
U 006A, DB,0015  ;4921          NOP                               ;NOT USED
U 006B, DB,0015  ;4922          NOP                               ;NOT USED
U 006C, DB,0015  ;4923          NOP                               ;NOT USED
U 006D, DB,0015  ;4924          NOP                               ;NOT USED
U 006E, DB,0015  ;4925          NOP                               ;NOT USED
U 006F, DB,0015  ;4926          NOP                               ;NOT USED
```

```
                    ;4927   .PAGE
                    ;4928   .REGION/70,5FF
                    ;4929   .TOC    "LS DATA SECTION"
                    ;4930
                    ;4931   ;+++
                    ;4932   ;       This section lists the data that will be transfered to LS by the con-
                    ;4933   ;       sole as part of the initialization performed in test 0.  The TRANSFER
                    ;4934   ;       DATA routine will point to this data area.  LS contains the constants,
                    ;4935   ;       control and status word, and temporary storage locations used by the
                    ;4936   ;       tests and subroutines of the microdiagnostic.
                    ;4937   ;         The LS is 32 bits wide.  Each of these words is 16 bits, so two con-
                    ;4938   ;       secutive words listed here will be loaded into each consecutive LS loc-
                    ;4939   ;       ation.  The first word listed will be bits 0-15 of the LS location, the
                    ;4940   ;       second word listed will be bits 16-31 of the same LS location.
                    ;4941   ;         The locations 78-7F in the first half of LS and F8-FF in the second
                    ;4942   ;       half of LS are not actual LS locations.  They force an access to one of
                    ;4943   ;       several registers in the CPU or force an indexing feature to another LS
                    ;4944   ;       location.  For that reason, they are not written via the transfer rou-
                    ;4945   ;       tine.
                    ;4946   ;         Note:  This table is in hexadecimal format i.e. each digit represents
                    ;4947   ;       four bits, so four digits represnts a full 16 bit word.  The micro-
                    ;4948   ;       assembler requires that a hexidecimal value that starts with a letter
                    ;4949   ;       (e.g. FFFF) be preceeded by a 0 (OFFFF).  So some table values will
                    ;4950   ;       contain 5 hexidecimal digits.  The fifth digit is not actually used.
                    ;4951   ;
                    ;4952   ;---
                    ;4953
                    ;4954   TRANSFER.DATA.LS1:
                    ;4955
U 0070, 00,0078     ;4956           COUNT.LS[0078]          ;LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANS-
                    ;4957                                   ; FER TO LS = 120 DECIMAL) DESTINATION IS LS
U 0071, 00,0000     ;4958           START.ADR[0000]         ;DESTINATION START ADDRESS (START AT LS 0)
                    ;4959
U 0072, 00,0000     ;4960           WORD[0000]              ;LOCATION 0
U 0073, 00,0000     ;4961           WORD[0000]
                    ;4962
U 0074, 00,0000     ;4963           WORD[0000]              ;LOCATION 1
U 0075, 00,0000     ;4964           WORD[0000]
                    ;4965
U 0076, 00,0000     ;4966           WORD[0000]              ;LOCATION 2
U 0077, 00,0000     ;4967           WORD[0000]
                    ;4968
U 0078, 00,0000     ;4969           WORD[0000]              ;LOCATION 3
U 0079, 00,0000     ;4970           WORD[0000]
                    ;4971
U 007A, 00,0000     ;4972           WORD[0000]              ;LOCATION 4
U 007B, 00,0000     ;4973           WORD[0000]
                    ;4974
U 007C, 00,0000     ;4975           WORD[0000]              ;LOCATION 5
U 007D, 00,0000     ;4976           WORD[0000]
                    ;4977
U 007E, 00,0000     ;4978           WORD[0000]              ;LOCATION 6
U 007F, 00,0000     ;4979           WORD[0000]
                    ;4980
U 0080, 00,0000     ;4981           WORD[0000]              ;LOCATION 7
```

```
U 0081, 00,0000   ;4982          WORD[0000]
                  ;4983
U 0082, 00,0000   ;4984          WORD[0000]              ;LOCATION 8
U 0083, 00,0000   ;4985          WORD[0000]
                  ;4986
U 0084, 00,0000   ;4987          WORD[0000]              ;LOCATION 9
U 0085, 00,0000   ;4988          WORD[0000]
                  ;4989
U 0086, 00,0000   ;4990          WORD[0000]              ;LOCATION 0A
U 0087, 00,0000   ;4991          WORD[0000]
                  ;4992
U 0088, 00,0000   ;4993          WORD[0000]              ;LOCATION 0B
U 0089, 00,0000   ;4994          WORD[0000]
                  ;4995
U 008A, 00,0000   ;4996          WORD[0000]              ;LOCATION 0C
U 008B, 00,0000   ;4997          WORD[0000]
                  ;4998
U 008C, 00,0000   ;4999          WORD[0000]              ;LOCATION 0D
U 008D, 00,0000   ;5000          WORD[0000]
                  ;5001
U 008E, 00,0000   ;5002          WORD[0000]              ;LOCATION 0E
U 008F, 00,0000   ;5003          WORD[0000]
                  ;5004
U 0090, 00,0000   ;5005          WORD[0000]              ;LOCATION 0F
U 0091, 00,0000   ;5006          WORD[0000]
                  ;5007
U 0092, 00,0000   ;5008          WORD[0000]              ;LOCATION 10
U 0093, 00,0000   ;5009          WORD[0000]
                  ;5010
U 0094, 00,0000   ;5011          WORD[0000]              ;LOCATION 11
U 0095, 00,0000   ;5012          WORD[0000]
                  ;5013
U 0096, 00,0000   ;5014          WORD[0000]              ;LOCATION 12
U 0097, 00,0000   ;5015          WORD[0000]
                  ;5016
U 0098, 00,00FF   ;5017          WORD[00FF]              ;LOCATION 13
U 0099, 00,0000   ;5018          WORD[0000]
                  ;5019
U 009A, 00,FFFF   ;5020          WORD[0FFFF]             ;LOCATION 14
U 009B, 00,0000   ;5021          WORD[0000]
                  ;5022
U 009C, 00,0000   ;5023          WORD[0000]              ;LOCATION 15
U 009D, 00,FF00   ;5024          WORD[0FF00]
                  ;5025
U 009E, 00,FF00   ;5026          WORD[0FF00]             ;LOCATION 16
U 009F, 00,FFFF   ;5027          WORD[0FFFF]
                  ;5028
U 00A0, 00,3FFF   ;5029          WORD[3FFF]              ;LOCATION 17
U 00A1, 00,0100   ;5030          WORD[0100]
                  ;5031
U 00A2, 00,3FFF   ;5032          WORD[3FFF]              ;LOCATION 18
U 00A3, 00,7FFE   ;5033          WORD[7FFE]
                  ;5034
U 00A4, 00,FFFF   ;5035          WORD[0FFFF]             ;LOCATION 19
U 00A5, 00,FE7F   ;5036          WORD[0FE7F]
```

```
                    ;5037
U 00A6, 00,0000     ;5038           WORD[0000]              ;LOCATION 1A
U 00A7, 00,7FF8     ;5039           WORD[7FF8]
                    ;5040
U 00A8, 00,8000     ;5041           WORD[8000]              ;LOCATION 1B
U 00A9, 00,7FFF     ;5042           WORD[7FFF]
                    ;5043
U 00AA, 00,0000     ;5044           WORD[0000]              ;LOCATION 1C
U 00AB, 00,0000     ;5045           WORD[0000]
                    ;5046
U 00AC, 00,0000     ;5047           WORD[0000]              ;LOCATION 1D
U 00AD, 00,0000     ;5048           WORD[0000]
                    ;5049
U 00AE, 00,0500     ;5050           WORD[0500]              ;LOCATION 1E
U 00AF, 00,0080     ;5051           WORD[0080]
                    ;5052
U 00B0, 00,0500     ;5053           WORD[0500]              ;LOCATION 1F
U 00B1, 00,0000     ;5054           WORD[0000]
                    ;5055
U 00B2, 00,0001     ;5056           WORD[0001]              ;LOCATION 20
U 00B3, 00,0000     ;5057           WORD[0000]
                    ;5058
U 00B4, 00,0002     ;5059           WORD[0002]              ;LOCATION 21
U 00B5, 00,0000     ;5060           WORD[0000]
                    ;5061
U 00B6, 00,0004     ;5062           WORD[0004]              ;LOCATION 22
U 00B7, 00,0000     ;5063           WORD[0000]
                    ;5064
U 00B8, 00,0008     ;5065           WORD[0008]              ;LOCATION 23
U 00B9, 00,0000     ;5066           WORD[0000]
                    ;5067
U 00BA, 00,0010     ;5068           WORD[0010]              ;LOCATION 24
U 00BB, 00,0000     ;5069           WORD[0000]
                    ;5070
U 00BC, 00,0020     ;5071           WORD[0020]              ;LOCATION 25
U 00BD, 00,0000     ;5072           WORD[0000]
                    ;5073
U 00BE, 00,0040     ;5074           WORD[0040]              ;LOCATION 26
U 00BF, 00,0000     ;5075           WORD[0000]
                    ;5076
U 00C0, 00,0080     ;5077           WORD[0080]              ;LOCATION 27
U 00C1, 00,0000     ;5078           WORD[0000]
                    ;5079
U 00C2, 00,0100     ;5080           WORD[0100]              ;LOCATION 28
U 00C3, 00,0000     ;5081           WORD[0000]
                    ;5082
U 00C4, 00,0200     ;5083           WORD[0200]              ;LOCATION 29
U 00C5, 00,0000     ;5084           WORD[0000]
                    ;5085
U 00C6, 00,0400     ;5086           WORD[0400]              ;LOCATION 2A
U 00C7, 00,0000     ;5087           WORD[0000]
                    ;5088
U 00C8, 00,0800     ;5089           WORD[0800]              ;LOCATION 2B
U 00C9, 00,0000     ;5090           WORD[0000]
                    ;5091
```

```
U 00CA, 00,1000   ;5092          WORD[1000]          ;LOCATION 2C
U 00CB, 00,0000   ;5093          WORD[0000]
                  ;5094
U 00CC, 00,2000   ;5095          WORD[2000]          ;LOCATION 2D
U 00CD, 00,0000   ;5096          WORD[0000]
                  ;5097
U 00CE, 00,4000   ;5098          WORD[4000]          ;LOCATION 2E
U 00CF, 00,0000   ;5099          WORD[0000]
                  ;5100
U 00D0, 00,8000   ;5101          WORD[8000]          ;LOCATION 2F
U 00D1, 00,0000   ;5102          WORD[0000]
                  ;5103
U 00D2, 00,0000   ;5104          WORD[0000]          ;LOCATION 30
U 00D3, 00,0001   ;5105          WORD[0001]
                  ;5106
U 00D4, 00,0000   ;5107          WORD[0000]          ;LOCATION 31
U 00D5, 00,0002   ;5108          WORD[0002]
                  ;5109
U 00D6, 00,0000   ;5110          WORD[0000]          ;LOCATION 32
U 00D7, 00,0004   ;5111          WORD[0004]
                  ;5112
U 00D8, 00,0000   ;5113          WORD[0000]          ;LOCATION 33
U 00D9, 00,0008   ;5114          WORD[0008]
                  ;5115
U 00DA, 00,0000   ;5116          WORD[0000]          ;LOCATION 34
U 00DB, 00,0010   ;5117          WORD[0010]
                  ;5118
U 00DC, 00,0000   ;5119          WORD[0000]          ;LOCATION 35
U 00DD, 00,0020   ;5120          WORD[0020]
                  ;5121
U 00DE, 00,0000   ;5122          WORD[0000]          ;LOCATION 36
U 00DF, 00,0040   ;5123          WORD[0040]
                  ;5124
U 00E0, 00,0000   ;5125          WORD[0000]          ;LOCATION 37
U 00E1, 00,0080   ;5126          WORD[0080]
                  ;5127
U 00E2, 00,0000   ;5128          WORD[0000]          ;LOCATION 38
U 00E3, 00,0100   ;5129          WORD[0100]
                  ;5130
U 00E4, 00,0000   ;5131          WORD[0000]          ;LOCATION 39
U 00E5, 00,0200   ;5132          WORD[0200]
                  ;5133
U 00E6, 00,0000   ;5134          WORD[0000]          ;LOCATION 3A
U 00E7, 00,0400   ;5135          WORD[0400]
                  ;5136
U 00E8, 00,0000   ;5137          WORD[0000]          ;LOCATION 3B
U 00E9, 00,0800   ;5138          WORD[0800]
                  ;5139
U 00EA, 00,0000   ;5140          WORD[0000]          ;LOCATION 3C
U 00EB, 00,1000   ;5141          WORD[1000]
                  ;5142
U 00EC, 00,0000   ;5143          WORD[0000]          ;LOCATION 3D
U 00ED, 00,2000   ;5144          WORD[2000]
                  ;5145
U 00EE, 00,0000   ;5146          WORD[0000]          ;LOCATION 3E
```

```
U 00EF, 00,4000  ;5147          WORD[4000]
                 ;5148
U 00F0, 00,0000  ;5149          WORD[0000]              ;LOCATION 3F
U 00F1, 00,8000  ;5150          WORD[8000]
                 ;5151
U 00F2, 00,0000  ;5152          WORD[0000]              ;LOCATION 40
U 00F3, 00,0000  ;5153          WORD[0000]
                 ;5154
U 00F4, 00,0000  ;5155          WORD[0000]              ;LOCATION 41
U 00F5, 00,0000  ;5156          WORD[0000]
                 ;5157
U 00F6, 00,0000  ;5158          WORD[0000]              ;LOCATION 42
U 00F7, 00,0000  ;5159          WORD[0000]
                 ;5160
U 00F8, 00,0000  ;5161          WORD[0000]              ;LOCATION 43
U 00F9, 00,0000  ;5162          WORD[0000]
                 ;5163
U 00FA, 00,0000  ;5164          WORD[0000]              ;LOCATION 44
U 00FB, 00,0000  ;5165          WORD[0000]
                 ;5166
U 00FC, 00,0000  ;5167          WORD[0000]              ;LOCATION 45
U 00FD, 00,0000  ;5168          WORD[0000]
                 ;5169
U 00FE, 00,0000  ;5170          WORD[0000]              ;LOCATION 46
U 00FF, 00,0000  ;5171          WORD[0000]
                 ;5172
U 0100, 00,0000  ;5173          WORD[0000]              ;LOCATION 47
U 0101, 00,0000  ;5174          WORD[0000]
                 ;5175
U 0102, 00,0000  ;5176          WORD[0000]              ;LOCATION 48
U 0103, 00,0000  ;5177          WORD[0000]
                 ;5178
U 0104, 00,0000  ;5179          WORD[0000]              ;LOCATION 49
U 0105, 00,0000  ;5180          WORD[0000]
                 ;5181
U 0106, 00,0000  ;5182          WORD[0000]              ;LOCATION 4A
U 0107, 00,0000  ;5183          WORD[0000]
                 ;5184
U 0108, 00,0000  ;5185          WORD[0000]              ;LOCATION 4B
U 0109, 00,0000  ;5186          WORD[0000]
                 ;5187
U 010A, 00,AAAA  ;5188          WORD[0AAAA]             ;LOCATION 4C
U 010B, 00,AAAA  ;5189          WORD[0AAAA]
                 ;5190
U 010C, 00,5555  ;5191          WORD[5555]              ;LOCATION 4D
U 010D, 00,5555  ;5192          WORD[5555]
                 ;5193
U 010E, 00,0000  ;5194          WORD[0000]              ;LOCATION 4E
U 010F, 00,0000  ;5195          WORD[0000]
                 ;5196
U 0110, 00,FFFF  ;5197          WORD[0FFFF]             ;LOCATION 4F
U 0111, 00,FFFF  ;5198          WORD[0FFFF]
                 ;5199
U 0112, 00,0000  ;5200          WORD[0000]              ;LOCATION 50
U 0113, 00,0000  ;5201          WORD[0000]
```

```
                      ;5202
U 0114, 00,0000       ;5203         WORD[0000]            ;LOCATION 51
U 0115, 00,0000       ;5204         WORD[0000]
                      ;5205
U 0116, 00,0000       ;5206         WORD[0000]            ;LOCATION 52
U 0117, 00,0000       ;5207         WORD[0000]
                      ;5208
U 0118, 00,0000       ;5209         WORD[0000]            ;LOCATION 53
U 0119, 00,0000       ;5210         WORD[0000]
                      ;5211
U 011A, 00,0000       ;5212         WORD[0000]            ;LOCATION 54
U 011B, 00,0000       ;5213         WORD[0000]
                      ;5214
U 011C, 00,0000       ;5215         WORD[0000]            ;LOCATION 55
U 011D, 00,0000       ;5216         WORD[0000]
                      ;5217
U 011E, 00,0000       ;5218         WORD[0000]            ;LOCATION 56
U 011F, 00,0000       ;5219         WORD[0000]
                      ;5220
U 0120, 00,0000       ;5221         WORD[0000]            ;LOCATION 57
U 0121, 00,0000       ;5222         WORD[0000]
                      ;5223
U 0122, 00,0000       ;5224         WORD[0000]            ;LOCATION 58
U 0123, 00,0000       ;5225         WORD[0000]
                      ;5226
U 0124, 00,0000       ;5227         WORD[0000]            ;LOCATION 59
U 0125, 00,0000       ;5228         WORD[0000]
                      ;5229
U 0126, 00,F000       ;5230         WORD[0F000]           ;LOCATION 5A
U 0127, 00,FFFF       ;5231         WORD[0FFFF]
                      ;5232
U 0128, 00,F002       ;5233         WORD[0F002]           ;LOCATION 5B
U 0129, 00,FFFF       ;5234         WORD[0FFFF]
                      ;5235
U 012A, 00,F004       ;5236         WORD[0F004]           ;LOCATION 5C
U 012B, 00,FFFF       ;5237         WORD[0FFFF]
                      ;5238
U 012C, 00,F006       ;5239         WORD[0F006]           ;LOCATION 5D
U 012D, 00,FFFF       ;5240         WORD[0FFFF]
                      ;5241
U 012E, 00,F008       ;5242         WORD[0F008]           ;LOCATION 5E
U 012F, 00,FFFF       ;5243         WORD[0FFFF]
                      ;5244
U 0130, 00,F00E       ;5245         WORD[0F00E]           ;LOCATION 5F
U 0131, 00,FFFF       ;5246         WORD[0FFFF]
                      ;5247
U 0132, 00,0003       ;5248         WORD[0003]            ;LOCATION 60
U 0133, 00,0000       ;5249         WORD[0000]
                      ;5250
U 0134, 00,0012       ;5251         WORD[0012]            ;LOCATION 61
U 0135, 00,0000       ;5252         WORD[0000]
                      ;5253
U 0136, 00,3333       ;5254         WORD[3333]            ;LOCATION 62
U 0137, 00,3333       ;5255         WORD[3333]
                      ;5256
```

```
U 0138, 00,0F0F   ;5257          WORD[0F0F]                  ;LOCATION 63
U 0139, 00,0F0F   ;5258          WORD[0F0F]
                  ;5259
U 013A, 00,00FF   ;5260          WORD[00FF]                  ;LOCATION 64
U 013B, 00,00FF   ;5261          WORD[00FF]
                  ;5262
U 013C, 00,0162   ;5263          WORD[0162]                  ;LOCATION 65
U 013D, 00,0000   ;5264          WORD[0000]
                  ;5265
U 013E, 00,0028   ;5266          WORD[0028]                  ;LOCATION 66
U 013F, 00,0000   ;5267          WORD[0000]
                  ;5268
U 0140, 00,000C   ;5269          WORD[000C]                  ;LOCATION 67
U 0141, 00,0000   ;5270          WORD[0000]
                  ;5271
U 0142, 00,0000   ;5272          WORD[0000]                  ;LOCATION 68
U 0143, 00,0000   ;5273          WORD[0000]
                  ;5274
U 0144, 00,0000   ;5275          WORD[0000]                  ;LOCATION 69
U 0145, 00,0000   ;5276          WORD[0000]
                  ;5277
U 0146, 00,0000   ;5278          WORD[0000]                  ;LOCATION 6A
U 0147, 00,0000   ;5279          WORD[0000]
                  ;5280
U 0148, 00,0000   ;5281          WORD[0000]                  ;LOCATION 6B
U 0149, 00,0000   ;5282          WORD[0000]
                  ;5283
U 014A, 00,0000   ;5284          WORD[0000]                  ;LOCATION 6C
U 014B, 00,0000   ;5285          WORD[0000]
                  ;5286
U 014C, 00,0000   ;5287          WORD[0000]                  ;LOCATION 6D
U 014D, 00,0000   ;5288          WORD[0000]
                  ;5289
U 014E, 00,0000   ;5290          WORD[0000]                  ;LOCATION 6E
U 014F, 00,0000   ;5291          WORD[0000]
                  ;5292
U 0150, 00,0000   ;5293          WORD[0000]                  ;LOCATION 6F
U 0151, 00,0000   ;5294          WORD[0000]
                  ;5295
U 0152, 00,0000   ;5296          WORD[0000]                  ;LOCATION 70
U 0153, 00,0000   ;5297          WORD[0000]
                  ;5298
U 0154, 00,0000   ;5299          WORD[0000]                  ;LOCATION 71
U 0155, 00,0000   ;5300          WORD[0000]
                  ;5301
U 0156, 00,0000   ;5302          WORD[0000]                  ;LOCATION 72
U 0157, 00,0000   ;5303          WORD[0000]
                  ;5304
U 0158, 00,0000   ;5305          WORD[0000]                  ;LOCATION 73
U 0159, 00,0000   ;5306          WORD[0000]
                  ;5307
U 015A, 00,0000   ;5308          WORD[0000]                  ;LOCATION 74
U 015B, 00,0000   ;5309          WORD[0000]
                  ;5310
U 015C, 00,0000   ;5311          WORD[0000]                  ;LOCATION 75
```

```
U 015D, 00,0000  ;5312          WORD[0000]
                 ;5313
U 015E, 00,0000  ;5314          WORD[0000]              ;LOCATION 76
U 015F, 00,0000  ;5315          WORD[0000]
                 ;5316
U 0160, 00,0000  ;5317          WORD[0000]              ;LOCATION 77
U 0161, 00,0000  ;5318          WORD[0000]
                 ;5319
                 ;5320  .TOC    "PROGRAM SPECIFIC DATA SECTION (LS 80-F7)"
                 ;5321  ;
                 ;5322  ;       This section represents the second half of LS.  It is only acceseble
                 ;5323  ;       with a MOV instruction, or one that uses the MOVE microinstruction for-
                 ;5324  ;       mat.  This section will be loaded in a seperate transfer.
                 ;5325  ;
                 ;5326  TRANSFER.DATA.LS2:
U 0162, 00,0078  ;5327          COUNT.LS[0078]          ;LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANS-
                 ;5328                                  ; FER TO LS = 120 DECIMAL) DESTINATION IS LS
U 0163, 00,0080  ;5329          START.ADR[0080]         ;DESTINATION START ADDRESS (START AT LS 80(H))
                 ;5330
U 0164, 00,0002  ;5331          WORD[0002]              ;LOCATION 80
U 0165, 00,0000  ;5332          WORD[0000]
                 ;5333
U 0166, 00,0004  ;5334          WORD[0004]              ;LOCATION 81
U 0167, 00,0000  ;5335          WORD[0000]
                 ;5336
U 0168, 00,0006  ;5337          WORD[0006]              ;LOCATION 82
U 0169, 00,0000  ;5338          WORD[0000]
                 ;5339
U 016A, 00,0008  ;5340          WORD[0008]              ;LOCATION 83
U 016B, 00,0000  ;5341          WORD[0000]
                 ;5342
U 016C, 00,000A  ;5343          WORD[000A]              ;LOCATION 84
U 016D, 00,0000  ;5344          WORD[0000]
                 ;5345
U 016E, 00,000C  ;5346          WORD[000C]              ;LOCATION 85
U 016F, 00,0000  ;5347          WORD[0000]
                 ;5348
U 0170, 00,0000  ;5349          WORD[0000]              ;LOCATION 86
U 0171, 00,0000  ;5350          WORD[0000]
                 ;5351
U 0172, 00,0000  ;5352          WORD[0000]              ;LOCATION 87
U 0173, 00,0000  ;5353          WORD[0000]
                 ;5354
U 0174, 00,0000  ;5355          WORD[0000]              ;LOCATION 88
U 0175, 00,0000  ;5356          WORD[0000]
                 ;5357
U 0176, 00,0000  ;5358          WORD[0000]              ;LOCATION 89
U 0177, 00,0000  ;5359          WORD[0000]
                 ;5360
U 0178, 00,4450  ;5361          WORD[4450]              ;LOCATION 8A
U 0179, 00,4352  ;5362          WORD[4352]
                 ;5363
U 017A, 00,4B30  ;5364          WORD[4B30]              ;LOCATION 8B
U 017B, 00,4143  ;5365          WORD[4143]
                 ;5366
```

```
U 017C, 00,2020  ;5367          WORD[2020]                ;LOCATION 8C
U 017D, 00,3120  ;5368          WORD[3120]
                 ;5369
U 017E, 00,001E  ;5370          WORD[001E]                ;LOCATION 8D
U 017F, 00,0000  ;5371          WORD[0000]
                 ;5372
U 0180, 00,0000  ;5373          WORD[0000]                ;LOCATION 8E
U 0181, 00,0000  ;5374          WORD[0000]
                 ;5375
U 0182, 00,0000  ;5376          WORD[0000]                ;LOCATION 8F
U 0183, 00,0000  ;5377          WORD[0000]
                 ;5378
U 0184, 00,0000  ;5379          WORD[0000]                ;LOCATION 90
U 0185, 00,0000  ;5380          WORD[0000]
                 ;5381
U 0186, 00,4450  ;5382          WORD[4450]                ;LOCATION 91
U 0187, 00,4352  ;5383          WORD[4352]
                 ;5384
U 0188, 00,4B20  ;5385          WORD[4B20]                ;LOCATION 92
U 0189, 00,4143  ;5386          WORD[4143]
                 ;5387
U 018A, 00,0000  ;5388          WORD[0000]                ;LOCATION 93
U 018B, 00,0000  ;5389          WORD[0000]
                 ;5390
U 018C, 00,5241  ;5391          WORD[5241]                ;LOCATION 94
U 018D, 00,5343  ;5392          WORD[5343]
                 ;5393
U 018E, 00,4820  ;5394          WORD[4820]                ;LOCATION 95
U 018F, 00,5443  ;5395          WORD[5443]
                 ;5396
U 0190, 00,2020  ;5397          WORD[2020]                ;LOCATION 96
U 0191, 00,2020  ;5398          WORD[2020]
                 ;5399
U 0192, 00,001F  ;5400          WORD[001F]                ;LOCATION 97
U 0193, 00,0000  ;5401          WORD[0000]
                 ;5402
U 0194, 00,001A  ;5403          WORD[001A]                ;LOCATION 98
U 0195, 00,0000  ;5404          WORD[0000]
                 ;5405
U 0196, 00,0000  ;5406          WORD[0000]                ;LOCATION 99
U 0197, 00,0000  ;5407          WORD[0000]
                 ;5408
U 0198, 00,0000  ;5409          WORD[0000]                ;LOCATION 9A
U 0199, 00,0000  ;5410          WORD[0000]
                 ;5411
U 019A, 00,0000  ;5412          WORD[0000]                ;LOCATION 9B
U 019B, 00,0000  ;5413          WORD[0000]
                 ;5414
U 019C, 00,0000  ;5415          WORD[0000]                ;LOCATION 9C
U 019D, 00,0000  ;5416          WORD[0000]
                 ;5417
U 019E, 00,0000  ;5418          WORD[0000]                ;LOCATION 9D
U 019F, 00,0000  ;5419          WORD[0000]
                 ;5420
U 01A0, 00,0000  ;5421          WORD[0000]                ;LOCATION 9E
```

```
U 01A1, 00,0000   ;5422          WORD[0000]
                  ;5423
U 01A2, 00,0000   ;5424          WORD[0000]          ;LOCATION 9F
U 01A3, 00,0000   ;5425          WORD[0000]
                  ;5426
U 01A4, 00,0000   ;5427          WORD[0000]          ;LOCATION 0A0
U 01A5, 00,0000   ;5428          WORD[0000]
                  ;5429
U 01A6, 00,0000   ;5430          WORD[0000]          ;LOCATION 0A1
U 01A7, 00,0000   ;5431          WORD[0000]
                  ;5432
U 01A8, 00,0000   ;5433          WORD[0000]          ;LOCATION 0A2
U 01A9, 00,0000   ;5434          WORD[0000]
                  ;5435
U 01AA, 00,0000   ;5436          WORD[0000]          ;LOCATION 0A3
U 01AB, 00,0000   ;5437          WORD[0000]
                  ;5438
U 01AC, 00,0000   ;5439          WORD[0000]          ;LOCATION 0A4
U 01AD, 00,0000   ;5440          WORD[0000]
                  ;5441
U 01AE, 00,0000   ;5442          WORD[0000]          ;LOCATION 0A5
U 01AF, 00,0000   ;5443          WORD[0000]
                  ;5444
U 01B0, 00,0000   ;5445          WORD[0000]          ;LOCATION 0A6
U 01B1, 00,0000   ;5446          WORD[0000]
                  ;5447
U 01B2, 00,0000   ;5448          WORD[0000]          ;LOCATION 0A7
U 01B3, 00,0000   ;5449          WORD[0000]
                  ;5450
U 01B4, 00,0000   ;5451          WORD[0000]          ;LOCATION 0A8
U 01B5, 00,0000   ;5452          WORD[0000]
                  ;5453
U 01B6, 00,0000   ;5454          WORD[0000]          ;LOCATION 0A9
U 01B7, 00,0000   ;5455          WORD[0000]
                  ;5456
U 01B8, 00,0000   ;5457          WORD[0000]          ;LOCATION 0AA
U 01B9, 00,0000   ;5458          WORD[0000]
                  ;5459
U 01BA, 00,0000   ;5460          WORD[0000]          ;LOCATION 0AB
U 01BB, 00,0000   ;5461          WORD[0000]
                  ;5462
U 01BC, 00,0000   ;5463          WORD[0000]          ;LOCATION 0AC
U 01BD, 00,0000   ;5464          WORD[0000]
                  ;5465
U 01BE, 00,0000   ;5466          WORD[0000]          ;LOCATION 0AD
U 01BF, 00,0000   ;5467          WORD[0000]
                  ;5468
U 01C0, 00,0000   ;5469          WORD[0000]          ;LOCATION 0AE
U 01C1, 00,0000   ;5470          WORD[0000]
                  ;5471
U 01C2, 00,0000   ;5472          WORD[0000]          ;LOCATION 0AF
U 01C3, 00,0000   ;5473          WORD[0000]
                  ;5474
U 01C4, 00,0000   ;5475          WORD[0000]          ;LOCATION 0B0
U 01C5, 00,0000   ;5476          WORD[0000]
```

```
                      ;5477
U 01C6, 00,0000       ;5478         WORD[0000]               ;LOCATION OB1
U 01C7, 00,0000       ;5479         WORD[0000]
                      ;5480
U 01C8, 00,0000       ;5481         WORD[0000]               ;LOCATION OB2
U 01C9, 00,0000       ;5482         WORD[0000]
                      ;5483
U 01CA, 00,0000       ;5484         WORD[0000]               ;LOCATION OB3
U 01CB, 00,0000       ;5485         WORD[0000]
                      ;5486
U 01CC, 00,0000       ;5487         WORD[0000]               ;LOCATION OB4
U 01CD, 00,0000       ;5488         WORD[0000]
                      ;5489
U 01CE, 00,0000       ;5490         WORD[0000]               ;LOCATION OB5
U 01CF, 00,0000       ;5491         WORD[0000]
                      ;5492
U 01D0, 00,0000       ;5493         WORD[0000]               ;LOCATION OB6
U 01D1, 00,0000       ;5494         WORD[0000]
                      ;5495
U 01D2, 00,0000       ;5496         WORD[0000]               ;LOCATION OB7
U 01D3, 00,0000       ;5497         WORD[0000]
                      ;5498
U 01D4, 00,0000       ;5499         WORD[0000]               ;LOCATION OB8
U 01D5, 00,0000       ;5500         WORD[0000]
                      ;5501
U 01D6, 00,0000       ;5502         WORD[0000]               ;LOCATION OB9
U 01D7, 00,0000       ;5503         WORD[0000]
                      ;5504
U 01D8, 00,0000       ;5505         WORD[0000]               ;LOCATION OBA
U 01D9, 00,0000       ;5506         WORD[0000]
                      ;5507
U 01DA, 00,0000       ;5508         WORD[0000]               ;LOCATION OBB
U 01DB, 00,0000       ;5509         WORD[0000]
                      ;5510
U 01DC, 00,0000       ;5511         WORD[0000]               ;LOCATION OBC
U 01DD, 00,0000       ;5512         WORD[0000]
                      ;5513
U 01DE, 00,0000       ;5514         WORD[0000]               ;LOCATION OBD
U 01DF, 00,0000       ;5515         WORD[0000]
                      ;5516
U 01E0, 00,0000       ;5517         WORD[0000]               ;LOCATION OBE
U 01E1, 00,0000       ;5518         WORD[0000]
                      ;5519
U 01E2, 00,0000       ;5520         WORD[0000]               ;LOCATION OBF
U 01E3, 00,0000       ;5521         WORD[0000]
                      ;5522
U 01E4, 00,0000       ;5523         WORD[0000]               ;LOCATION OC0
U 01E5, 00,0000       ;5524         WORD[0000]
                      ;5525
U 01E6, 00,4E00       ;5526         WORD[4E00]               ;LOCATION OC1
U 01E7, 00,0006       ;5527         WORD[0006]
                      ;5528
U 01E8, 00,0000       ;5529         WORD[0000]               ;LOCATION OC2
U 01E9, 00,0000       ;5530         WORD[0000]
                      ;5531
```

```
U 01EA, 00,0000   ;5532         WORD[0000]              ;LOCATION 0C3
U 01EB, 00,0000   ;5533         WORD[0000]
                  ;5534
U 01EC, 00,0000   ;5535         WORD[0000]              ;LOCATION 0C4
U 01ED, 00,0000   ;5536         WORD[0000]
                  ;5537
U 01EE, 00,0000   ;5538         WORD[0000]              ;LOCATION 0C5
U 01EF, 00,0000   ;5539         WORD[0000]
                  ;5540
U 01F0, 00,0000   ;5541         WORD[0000]              ;LOCATION 0C6
U 01F1, 00,0000   ;5542         WORD[0000]
                  ;5543
U 01F2, 00,0000   ;5544         WORD[0000]              ;LOCATION 0C7
U 01F3, 00,0000   ;5545         WORD[0000]
                  ;5546
U 01F4, 00,4E00   ;5547         WORD[4E00]              ;LOCATION 0C8
U 01F5, 00,0006   ;5548         WORD[0006]
                  ;5549
U 01F6, 00,0005   ;5550         WORD[0005]              ;LOCATION 0C9
U 01F7, 00,0000   ;5551         WORD[0000]
                  ;5552
U 01F8, 00,0006   ;5553         WORD[0006]              ;LOCATION 0CA
U 01F9, 00,0000   ;5554         WORD[0000]
                  ;5555
U 01FA, 00,0007   ;5556         WORD[0007]              ;LOCATION 0CB
U 01FB, 00,0000   ;5557         WORD[0000]
                  ;5558
U 01FC, 00,0026   ;5559         WORD[0026]              ;LOCATION 0CC
U 01FD, 00,0000   ;5560         WORD[0000]
                  ;5561
U 01FE, 00,FFF8   ;5562         WORD[0FFF8]             ;LOCATION 0CD
U 01FF, 00,FFFF   ;5563         WORD[0FFFF]
                  ;5564
U 0200, 00,FF80   ;5565         WORD[0FF80]             ;LOCATION 0CE
U 0201, 00,FFFF   ;5566         WORD[0FFFF]
                  ;5567
U 0202, 00,FFFE   ;5568         WORD[0FFFE]             ;LOCATION 0CF
U 0203, 00,FBFF   ;5569         WORD[0FBFF]
                  ;5570
U 0204, 00,20C7   ;5571         WORD[20C7]              ;LOCATION 0D0
U 0205, 00,FFFF   ;5572         WORD[0FFFF]
                  ;5573
U 0206, 00,007F   ;5574         WORD[007F]              ;LOCATION 0D1
U 0207, 00,FFFF   ;5575         WORD[0FFFF]
                  ;5576
U 0208, 00,0080   ;5577         WORD[0080]              ;LOCATION 0D2
U 0209, 00,FFFF   ;5578         WORD[0FFFF]
                  ;5579
U 020A, 00,FFFF   ;5580         WORD[0FFFF]             ;LOCATION 0D3
U 020B, 00,0000   ;5581         WORD[0000]
                  ;5582
U 020C, 00,0018   ;5583         WORD[0018]              ;LOCATION 0D4
U 020D, 00,0000   ;5584         WORD[0000]
                  ;5585
U 020E, 00,0001   ;5586         WORD[0001]              ;LOCATION 0D5
```

```
U 020F, 00,0400   ;5587          WORD[0400]
                  ;5588
U 0210, 00,6000   ;5589          WORD[6000]              ;LOCATION 0D6
U 0211, 00,0004   ;5590          WORD[0004]
                  ;5591
U 0212, 00,000F   ;5592          WORD[000F]              ;LOCATION 0D7
U 0213, 00,0000   ;5593          WORD[0000]
                  ;5594
U 0214, 00,FFE0   ;5595          WORD[0FFE0]             ;LOCATION 0D8
U 0215, 00,FFFF   ;5596          WORD[0FFFF]
                  ;5597
U 0216, 00,FFC0   ;5598          WORD[0FFC0]             ;LOCATION 0D9
U 0217, 00,FFFF   ;5599          WORD[0FFFF]
                  ;5600
U 0218, 00,FC00   ;5601          WORD[0FC00]             ;LOCATION 0DA
U 0219, 00,FFFF   ;5602          WORD[0FFFF]
                  ;5603
U 021A, 00,0230   ;5604          WORD[0230]              ;LOCATION 0DB
U 021B, 00,0000   ;5605          WORD[0000]
                  ;5606
U 021C, 00,FFFE   ;5607          WORD[0FFFE]             ;LOCATION 0DC
U 021D, 00,FFFF   ;5608          WORD[0FFFF]
                  ;5609
U 021E, 00,FFEF   ;5610          WORD[0FFEF]             ;LOCATION 0DD
U 021F, 00,FFFF   ;5611          WORD[0FFFF]
                  ;5612
U 0220, 00,C0FF   ;5613          WORD[0C0FF]             ;LOCATION 0DE
U 0221, 00,FFFF   ;5614          WORD[0FFFF]
                  ;5615
U 0222, 00,1A00   ;5616          WORD[1A00]              ;LOCATION 0DF
U 0223, 00,0000   ;5617          WORD[0000]
                  ;5618
U 0224, 00,0C00   ;5619          WORD[0C00]              ;LOCATION 0E0
U 0225, 00,0000   ;5620          WORD[0000]
                  ;5621
U 0226, 00,C000   ;5622          WORD[0C000]             ;LOCATION 0E1
U 0227, 00,0000   ;5623          WORD[0000]
                  ;5624
U 0228, 00,000B   ;5625          WORD[000B]              ;LOCATION 0E2
U 0229, 00,0000   ;5626          WORD[0000]
                  ;5627
U 022A, 00,001D   ;5628          WORD[001D]              ;LOCATION 0E3
U 022B, 00,0000   ;5629          WORD[0000]
                  ;5630
U 022C, 00,00FF   ;5631          WORD[00FF]              ;LOCATION 0E4
U 022D, 00,FFFF   ;5632          WORD[0FFFF]
                  ;5633
U 022E, 00,0000   ;5634          WORD[0000]              ;LOCATION 0E5
U 022F, 00,0000   ;5635          WORD[0000]
                  ;5636
U 0230, 00,0000   ;5637          WORD[0000]              ;LOCATION 0E6
U 0231, 00,0000   ;5638          WORD[0000]
                  ;5639
U 0232, 00,0000   ;5640          WORD[0000]              ;LOCATION 0E7
U 0233, 00,0000   ;5641          WORD[0000]
```

```
                  ;5642
U 0234, 00,0000   ;5643          WORD[0000]                  ;LOCATION 0E8
U 0235, 00,0000   ;5644          WORD[0000]
                  ;5645
U 0236, 00,0000   ;5646          WORD[0000]                  ;LOCATION 0E9
U 0237, 00,0000   ;5647          WORD[0000]
                  ;5648
U 0238, 00,0000   ;5649          WORD[0000]                  ;LOCATION 0EA
U 0239, 00,0000   ;5650          WORD[0000]
                  ;5651
U 023A, 00,0000   ;5652          WORD[0000]                  ;LOCATION 0EB
U 023B, 00,0000   ;5653          WORD[0000]
                  ;5654
U 023C, 00,0000   ;5655          WORD[0000]                  ;LOCATION 0EC
U 023D, 00,0000   ;5656          WORD[0000]
                  ;5657
U 023E, 00,0000   ;5658          WORD[0000]                  ;LOCATION 0ED
U 023F, 00,0000   ;5659          WORD[0000]
                  ;5660
U 0240, 00,0000   ;5661          WORD[0000]                  ;LOCATION 0EE
U 0241, 00,0000   ;5662          WORD[0000]
                  ;5663
U 0242, 00,0000   ;5664          WORD[0000]                  ;LOCATION 0EF
U 0243, 00,0000   ;5665          WORD[0000]
                  ;5666
U 0244, 00,0000   ;5667          WORD[0000]                  ;LOCATION 0F0
U 0245, 00,0000   ;5668          WORD[0000]
                  ;5669
U 0246, 00,0000   ;5670          WORD[0000]                  ;LOCATION 0F1
U 0247, 00,0000   ;5671          WORD[0000]
                  ;5672
U 0248, 00,0000   ;5673          WORD[0000]                  ;LOCATION 0F2
U 0249, 00,0000   ;5674          WORD[0000]
                  ;5675
U 024A, 00,0000   ;5676          WORD[0000]                  ;LOCATION 0F3
U 024B, 00,0000   ;5677          WORD[0000]
                  ;5678
U 024C, 00,0000   ;5679          WORD[0000]                  ;LOCATION 0F4
U 024D, 00,0000   ;5680          WORD[0000]
                  ;5681
U 024E, 00,00D8   ;5682          WORD[00D8]                  ;LOCATION 0F5
U 024F, 00,0000   ;5683          WORD[0000]
                  ;5684
U 0250, 00,FF80   ;5685          WORD[0FF80]                 ;LOCATION 0F6
U 0251, 00,0000   ;5686          WORD[0000]
                  ;5687
                  ;5688   .REGION/600,6FF
                  .PAGE    "WCS SUBROUTINES FOR CONSOLE USE"
                  .TOC     "  GENERATE TRANSFER DATA"
                  ;
                  ;        This routine is used by the diagnostic monitor to load WR 0 with a
                  ;        data pattern from the console.  The monitor will set CONSOLE ATTN if
                  ;        a 1 is to be generated, or clear CONSOLE ATTN if a 0 is to be gen-
                  ;        erated.  It will then single step this routine to shift the data bit
                  ;        into WR 0.  This routine loads a 32 bit value much more quickly than
                  ;        shifting each instruction into the CSR from the monitor, and is used
                  ;        mainly to set up data to load in LS for constants.
                  ;

                  ;                           ***WARNING***
                  ;
                  ;        This routine must start at 600 and end at 605.  DO NOT move this
                  ;        routine to any other area!!
                  ;
                  ;

                  GEN.XFER.DATA:
U 0600, 2F,8015            CLR WR[0]                       ;CLEAR A WORKING REG
U 0601, A0,C015            COM WR[0]                       ;NOW WR 0 IS ALL ONES
                  LOOP.XFER:
U 0602, 5B,0018            SKIP.IF[CONSOLE.ATTN]           ;SKIP NEXT INSTRUCTION IS CONSOLE ATTN
                                                           ; IS SET (GENERATING A 1)
                           ASHL WR[0],                     ;SHIFT A 0 INTO BIT 0 OF WR 0
U 0603, A3,D01E             SKIP                           ; AND SKIP ROL
U 0604, A3,C015            ROL WR[0]                       ;SHIFT A 1 INTO BIT 0 OF WR 0
U 0605, 08,6024            JMP [LOOP.XFER]                 ;REPEAT


                  ;
                  ;        DIAGNOSTIC VERSION NUMBER IS STORED HERE
                  ;

                  ;                           ***WARNING***
                  ;
                  ;        These words must be at location 606 and 607.  DO NOT move this word
                  ;        to any other area!!
                  ;
                  ;

U 0606, 00,0200            WORD[0200]                      ;VERSION 02.00
U 0607, 00,4347            ASCII [CG]                      ;LAST 2 LETTERS OF FILE NAME [ENKCG]


```

```
.PAGE   "  DEPOSIT MCT CSR ROUTINE"
;+++
;
; FUNCTIONAL DESCRIPTION:
;
;       The deposit to CSR routine is used to write the memory controller's
;       control and status register.  The memory controller has three CSRs
;       named CSR 0, CSR 1, and CSR 2.
;         CSR 0 contains checkbits or syndrome bits returned from the ECC logic
;       after certain diagnostic routines.  It is NOT writable directly with
;       this routine.
;         CSR 1 contains error bits set if an error occurs on a memory access.
;       It does contain five maintenance bits (bits 29-25) which are writable.
;       There are also seven checkbits (bits 6-0) which are writable, but not
;       readable.  These bits are used to write checkbits into the ECC chips.
;         CSR 2 contains error bits set if a UNIBUS error occurs on a UNIBUS
;       access.  These are read only bits and are NOT writable by this routine.
;         Therefore, CSR 1 is the only writable CSR, and only bits 29-25 can be
;       both written and read back.  This routine will thus force CSR 1 on a
;       deposit to CSR.
;         This routine will write LS 5 with data to access CSR 1, then write
;       the data residing in LS 6 into the CSR.  It will then issue CPU ATTN
;       to inform the console that it has completed the function.
;         NOTE:  The error bits of CSR 1 are cleared on any write to CSR 1.
;       These are bits 31, 30 and 23-14.Bits 13-0 and bit 24 are unused.
;
; CALLING SEQUENCE:
;
;       The console loads the address of a pointer to this routine (a JMP in-
;       struction at WCS location 50) into the UPC and starts the CPU.
;
; INPUT PARAMETERS:
;
;       LS location 6 must have been written by the console with the desired
;       data to be deposited in CSR 1 previous to executing this routine.
;
; IMPLICIT INPUTS:
;
;       None
;
; OUTPUT PARAMATERS:
;
;       CSR 1 bits 29-25 and 6-0 are written with data from LS 6.
;
; IMPLICIT OUTPUTS:
;
;       None
;
; SIDE EFFECTS:
;
;       WR 0 will be modified by this routine.
;       LS 5 will be modified by this routine.
;       CSR 1 bits 31, 30 and 23-14 are cleared.
;
;---
```

```
                        ;5790
                        ;5791   DEPOSIT.CSR:
                        ;5792
U 0608, 36,4415         ;5793           MOV LS[CSR1] TO WR[0]           ;SET BIT 2 IN WR 0 AS CSR NUMBER TO
                        ;5794                                           ; WRITE (BITS 2 AND 3 ARE CSR NUMBER)
U 0609, BE,0A15         ;5795           MOV WR[0] TO LS[T5.SUB]         ;PUT CSR NUMBER IN LS LOCATION 5 (CSR
                        ;5796                                           ; 1)
U 060A, 1D,0BF5         ;5797           MEM.REQ[WRITE.CSR] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                        ;5798                                           ; ACCESS CSR 1
U 060B, B2,0C15         ;5799           WRITE.MEM LS[T6.SUB]            ;SEND DATA IN LS LOCATION 6 TO CSR 1
U 060C, 88,6874         ;5800           JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
```

```
        .PAGE   "  EXAMINE MCT CSR ROUTINE"
        ;+++
        ;
        ; FUNCTIONAL DESCRIPTION:
        ;
        ;       The examine CSR routine reads the memory controller's control and
        ;       status register.  There are 3 CSRs labeled CSR 0, CSR 1 and CSR 2.
        ;         CSR 0 contains checkbits or syndrome bits from the ECC logic.
        ;       These are contained in bits 6-0 of the CSR.  Other bits are UNPRE-
        ;       DICTABLE.  Bits 6-0 are only written by special diagnostic routines
        ;       (they are not writtable by the DEPOSIT CSR command).
        ;         CSR 1 contains error bits and maintenance bits.  The error bits are
        ;       31, 30, and 23-14.  Maintenance bits are 29-25.  The other bits are
        ;       UNPREDICTABLE.  The error bits are written during memory functions
        ;       only and are not writable via the DEPOSIT CSR command. The maintenance
        ;       bits are writable.  Note that a DEPOSIT CSR command will write the
        ;       maintenance bits and clear all error bits.
        ;         CSR 2 contains three bits (bits 16-14) which are readable.  These are
        ;       set when a UNIBUS error occurs on a UNIBUS access.  They are not
        ;       directly writable with the DEPOSIT CSR command.
        ;         The console loads LS 5 with the CSR number (0, 1, or 2) to be read.
        ;       This routine rotates LS 5 two places left to position it correctly.
        ;       It then executes a read CSR and places the data in LS 6 for the console
        ;       to read.
        ;
        ; CALLING SEQUENCE:
        ;
        ;       The console loads the address of a pointer to this routine (a JMP in-
        ;       struction at WCS location 51) into the UPC and starts the CPU.
        ;
        ; INPUT PARAMETERS:
        ;
        ;       LS 5 contains the CSR number to be read.
        ;
        ; IMPLICIT INPUTS:
        ;
        ;       None
        ;
        ; OUTPUT PARAMATERS:
        ;
        ;       LS 6 - Data read from CSR selected.
        ;
        ; IMPLICIT OUTPUTS:
        ;
        ;       None
        ;
        ; SIDE EFFECTS:
        ;
        ;       LS 5 is rotated 2 places left.
        ;
        ;---

        EXAMINE.CSR:

U 060D, 36,0A15         MOV LS[T5.SUB] TO WR[0]         ;GET CSR NUMBER FROM LS 5
```

```
U 060E, A2,D015   ;5856           SHL2 WR[0]                              ;SHIFT NUMBER 2 PLACES LEFT TO GET
                  ;5857                                                   ; CSR NUMBER IN BITS 2 AND 3
U 060F, BE,0A15   ;5858           MOV WR[0] TO LS[T5.SUB]                 ;PUT SHIFTED NUMBER IN LS 5
                  ;5859
U 0610, 9D,0B75   ;5860           MEM.REQ[READ.CSR] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                  ;5861                                                   ; ACCESS CSR SPECIFIED
U 0611, 30,2215   ;5862           MOV MEM.DATA TO WR[0]                   ;READ CSR SELECTED
                  ;5863
                  ;5864   CHECK.CSR2:
U 0612, B6,0A95   ;5865           MOV LS[T5.SUB] TO WR[1]                 ;GET SHIFTED CSR NUMBER
                  ;5866           CMP LS[#8] WITH WR[1],                  ;SEE IF CSR 2 WAS SELECTED
U 0613, CE,46B5   ;5867            DT(LONG)&SET.ALU.CC                    ; SET UP CONDITION CODES
U 0614, 08,6161   ;5868           JMP.IF[NEQ] TO [CHECK.CSR1]             ;JUMP IF CSR 2 WAS NOT SELECTED
U 0615, C5,3015   ;5869           BIC LS[CSR2.MASK] TO WR[0]              ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
                  ;5870                                                   ; CSR 2.
                  ;5871   CHECK.CSR1:
                  ;5872           CMP LS[#4] WITH WR[1],                  ;SEE IF CSR 1 WAS SELECTED
U 0616, 4E,44B5   ;5873            DT(LONG)&SET.ALU.CC                    ; SET UP CONDITION CODES
U 0617, 08,6191   ;5874           JMP.IF[NEQ] TO [CHECK.CSR0]             ;JUMP IF CSR 1 WAS NOT SELECTED
U 0618, C5,2E15   ;5875           BIC LS[CSR1.MASK] TO WR[0]              ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
                  ;5876                                                   ; CSR 1.
                  ;5877   CHECK.CSR0:
                  ;5878           CMP LS[#0] WITH WR[1],                  ;SEE IF CSR 0 WAS SELECTED
U 0619, 4E,9CB5   ;5879            DT(LONG)&SET.ALU.CC                    ; SET UP CONDITION CODES
U 061A, 88,61D1   ;5880           JMP.IF[NEQ] TO [LOAD.LS6]               ;JUMP IF CSR 0 WAS NOT SELECTED
U 061B, 45,2C15   ;5881           BIC LS[CSR0.MASK] TO WR[0]              ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
                  ;5882                                                   ; CSR 0.
U 061C, C5,4E15   ;5883           BIC LS[BIT7] TO WR[0]                   ;BIT 7 IS ALSO UNUSED
                  ;5884
                  ;5885   LOAD.LS6:
U 061D, BE,0C15   ;5886           MOV WR[0] TO LS[T6.SUB]                 ;LOAD CSR DATA INTO LS 6 FOR PRINTOUT
U 061E, 88,6874   ;5887           JMP [WAIT.SUB]                          ;ISSUE CPU ATTN AND WAIT FOR RESPONSE
```

```
        .page   "  DEPOSIT MCT TRANSLATION BUFFER ROUTINE"
        ;+++
        ;
        ; FUNCTIONAL DESCRIPTION:
        ;
        ;       This routine allows a write to the translation buffer portion of the
        ;       memory controller.  The buffer area contains 128 decimal locations
        ;       addressed from 0-7F(H).  The remainder of the TB RAM is either unused
        ;       or contains the UNIBUS MAP.  The DEPOSIT UB command is used to write
        ;       the UNIBUS map portion of the TB RAMs.  The address specified with the
        ;       DEPOSIT command will be the actual RAM address accessed.  This routine
        ;       will do any shifting necessary on the address specified to position it
        ;       correctly.  The address specified has been loaded by the console in LS
        ;       location 5 prior to executing this routine.  The address must be in HEX
        ;       format.
        ;         The data specified has been placed in LS 6 by the console prior to
        ;       executing this routine.  Bits 07-01 will be the Valid, Prot, Modify,
        ;       and Byte Offset bits (there are four Prot bits).  Bits 22-08 will
        ;       contain the PA bits from PA 23 through PA 09 respectively.  Bits 31
        ;       through 23 and bit 0 are unused. This routine will do any shifting nec-
        ;       essary to write the bits in TB as described.  The data specified must
        ;       be in HEX format.
        ;
        ; CALLING SEQUENCE:
        ;
        ;       The console loads the address of a pointer to this routine (a JMP in-
        ;       struction at WCS location 52) into the UPC and starts the CPU.
        ;
        ; INPUT PARAMETERS:
        ;
        ;       LS 5 - TB address to be written (range 0-7F)
        ;       LS 6 - TB data to write (22 bits, from 1-22).  Bits 0 and 23-31 are
        ;               ignored.
        ;
        ; IMPLICIT INPUTS:
        ;
        ;       None
        ;
        ; OUTPUT PARAMATERS:
        ;
        ;       TB will be written with specified data at specified address
        ;
        ; IMPLICIT OUTPUTS:
        ;
        ;       None
        ;
        ; SIDE EFFECTS:
        ;
        ;       WR 0 will be modified
        ;       WR 1 will be modified
        ;
        ;---

        DEPOSIT.TB:
U 061F, 88,62FC         JSR [SHIFT.TB.ADR]                  ;GO TO SUBROUTINE TO POSITION TB AD-
```

```
                 ;5943                                          ; DRESS CORRECTLY
U 0620, 36,0C15  ;5944          MOV LS[T6.SUB] TO WR[0]         ;GET DATA FROM LS 6
U 0621, 45,2615  ;5945          BIC LS[#FF] TO WR[0]            ;CLEAR LOW BYTE OF DATA SPECIFIED.
                 ;5946                                          ; LOW BYTE WILL BE PUT IN BITS 24-31
                 ;5947                                          ; LATER
U 0622, 88,62AC  ;5948          JSR [SHIFT.LOOP]                ;GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
                 ;5949                                          ; PLACES RIGHT
U 0623, E4,0C15  ;5950          SWAP LS[T6.SUB] WITH WR[0]      ;SWAP ORIGINAL DATA SPECIFIED WITH MOD-
                 ;5951                                          ; IFIED DATA.  NOW LS 6 CONTAINS BITS
                 ;5952                                          ; 8-22 IN BITS 0-14 AS REQUIRED.  WR 0
                 ;5953                                          ; CONTAINS ORIGINAL DATA SPECIFIED
U 0624, 45,2C15  ;5954          BIC LS[#FFFFFF00] TO WR[0]      ;CLEAR ALL BUT LOW BYTE OF ORIGINAL
                 ;5955                                          ; DATA SPECIFIED
U 0625, 88,62AC  ;5956          JSR [SHIFT.LOOP]                ;GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
                 ;5957                                          ; PLACES RIGHT.  ON RETURN, WR 0 WILL
                 ;5958                                          ; HAVE BITS 0-7 IN BITS 24-31, OTHER
                 ;5959                                          ; BITS ARE CLEAR
U 0626, ED,0C15  ;5960          BIS WR[0] TO LS[T6.SUB]         ;NOW LS 6 CONTAINS DATA TO WRITE IN
                 ;5961                                          ; TB IN CORRECT FORMAT  I.E. PA BITS
                 ;5962                                          ; IN BITS 00-14 AND BYTE OFFSET, MODIFY
                 ;5963                                          ; PROT A THROUGH PROT D, AND VALID IN
                 ;5964                                          ; BITS 25-31.
                 ;5965
U 0627, 98,0AF5  ;5966          MEM.REQ[WRITE.TB] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO WRITE
                 ;5967                                          ; THE TB
U 0628, B2,0C15  ;5968          WRITE.MEM LS[T6.SUB]            ;WRITE DATA FROM LS 6 IN TB
U 0629, 88,6874  ;5969          JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
                 ;5970
                 ;5971   ;
                 ;5972   ;      SUBROUTINE TO SHIFT WR 0 TO THE RIGHT 8 PLACES
                 ;5973   ;
                 ;5974   SHIFT.LOOP:
U 062A, 36,4695  ;5975          MOV LS[#8] TO WR[1]             ;COUNTER IN WR 1 TO SHIFT 8 PLACES
                 ;5976   SHIFT.LOOP.1:
U 062B, 23,4015  ;5977          ROR WR[0]                       ;SHIFT WR 0 ONE PLACE RIGHT
                 ;5978          DEC WR[1],                      ;DECREMENT COUNTER
U 062C, A1,02B5  ;5979           DT(LONG)&SET.ALU.CC            ; AND SET CONDITION CODES
U 062D, 88,62B1  ;5980          JMP.IF[NEQ] TO [SHIFT.LOOP.1]   ;REPEAT UNTIL 8 SHIFTS HAVE OCCURRED
U 062E, 5B,0014  ;5981          RETURN                          ;THEN RETURN
                 ;5982   ;
                 ;5983   ;      SUBROUTINE TO POSITION TB ADDRESS IN BITS 9-14 AND BIT 31 (TB ADDRESS
                 ;5984   ;      BIT 0 MUST BE IN BIT 31).
                 ;5985   ;
                 ;5986   SHIFT.TB.ADR:
U 062F, 36,0A15  ;5987          MOV LS[T5.SUB] TO WR[0]         ;GET TB ADDRESS FROM LS 5
U 0630, 45,2C15  ;5988          BIC LS[#FFFFFF00] TO WR[0]      ;CLEAR ALL BUT LOW BYTE (8 BIT ADDRESS)
U 0631, A2,D015  ;5989          SHL2 WR[0]                      ;SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0632, A2,D015  ;5990          SHL2 WR[0]                      ;SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0633, A2,D015  ;5991          SHL2 WR[0]                      ;SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0634, A2,D015  ;5992          SHL2 WR[0]                      ;SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0635, 23,D015  ;5993          ASHL WR[0]                      ;SHIFT ONE MORE PLACE
                 ;5994                                          ; NOW BITS 0-6 IN BITS 9-15
                 ;5995          BIT LS[BIT15] WITH WR[0],       ;SEE IF BIT 15 (MSB) IS SET OR CLEAR.
U 0636, 59,5E35  ;5996           DT(LONG)&SET.ALU.CC            ; MSB MUST GO IN BIT 31 IN ORDER TO AD-
                 ;5997                                          ; DRESS TB CORRECTLY
```

```
U 0637, 5B,0009  ;5998          SKIP.IF[EQL]                        ;SKIP NEXT INSTRUCTION IF MSB IS 0
U 0638, 47,7E15  ;5999          BIS LS[BIT31] TO WR[0]              ;ELSE SET BIT 31 FOR MSB
                 ;6000
                 ;6001          MOV WR[0] TO LS[T5.SUB],            ;NOW TB ADDRESS IS IN CORRECT FORM IN
                 ;6002                                              ; LS LOCATION 5
U 0639, 3E,0A14  ;6003           RETURN                             ; THEN RETURN TO MAIN CODE
```

```
                        .PAGE   "  EXAMINE TRANSLATION BUFFER ROUTINE"
                        ;+++
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       This routine will read the translation buffer of the memory controller.
                        ;       The TB will be returned with the byte offset, modify, prot A through
                        ;       prot D, and valid in bits 01-07 respectively.  PA bits 09-23 are re-
                        ;       turned in bits 08-22 respectively.  Bits 23-31 and bit 0 are not used
                        ;       and so will be cleared before printing the result.  The result will be
                        ;       put in LS 6 for the console to print.
                        ;         The routine will issue a memory request with memory function=READ.TB
                        ;       and data type=LONGWORD.  The console will load LS 5 with the address
                        ;       specified before executing this routine.  The routine will shift the
                        ;       address around as required to address the TB location specified.  The
                        ;       address specified must be in HEX format.
                        ;         The TB consists of 128 decimal locations (addresses from 0-7F).  The
                        ;       remaining locations in the TB RAM are either unused or used by the
                        ;       UNIBUS map.  UNIBUS map addresses are read via the EXAMINE UB routine.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       The console loads the address of a pointer to this routine (a JMP in-
                        ;       struction at WCS location 53) into the UPC and starts the CPU.
                        ;
                        ; INPUT PARAMETERS:
                        ;
                        ;       LS 5 - TB address (0-7F) in hex format
                        ;
                        ; IMPLICIT INPUTS:
                        ;
                        ;       None
                        ;
                        ; OUTPUT PARAMATERS:
                        ;
                        ;       LS 6 - TB data from address specified
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;
                        ;       None
                        ;
                        ; SIDE EFFECTS:
                        ;
                        ;       LS 5 is modified
                        ;       WR 0 is modified
                        ;
                        ;---

                        EXAMINE.TB:
U 063A, 88,62FC                 JSR [SHIFT.TB.ADR]                      ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
                                                                        ; CORRECT POSITION AND REPLACE IN LS 5
U 063B, 9C,0BF5                 MEM.REQ[READ.TB] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO READ
                                                                        ; THE TB AT ADDRESS CONTAINED IN LS 5
U 063C, 30,2215                 MOV MEM.DATA TO WR[0]                   ;PLACE RESULT IN WR 0

```

M6

```
U 063D, 45,2A15   ;6059          BIC LS[#FF000000] TO WR[0]       ;CLEAR UPPER BYTE OF RESULT (UPPER BYTE
                  ;6060                                           ; IS NOT PART OF TB AND IS UNPREDICT-
                  ;6061                                           ; ABLE)
U 063E, 45,6E15   ;6062          BIC LS[BIT23] TO WR[0]           ;DITTO FOR BIT 23
U 063F, 45,4015   ;6063          BIC LS[BIT0] TO WR[0]            ;DITTO FOR BIT 0.  WR 0 NOW CONTAINS
                  ;6064                                           ; RESULT TO PRINT
U 0640, BE,0C15   ;6065          MOV WR[0] TO LS[T6.SUB]          ;RESULT NOW IN LS 6
U 0641, 88,6874   ;6066          JMP [WAIT.SUB]                   ;JUMP TO SET ATTN AND WAIT
```

```
                   .PAGE   "  DEPOSIT UNIBUS MAP ROUTINE"
                   ;+++
                   ;
                   ; FUNCTIONAL DESCRIPTION:
                   ;
                   ;         This routine allows a write to the UNIBUS map portion of the TB on the
                   ;         memory controller.  The map area contains 512 decimal locations
                   ;         addressed from 200-3FF.  The remainder of the TB RAM is either unused
                   ;         or contains the Translation Buffer information.  The DEPOSIT TB command
                   ;         is used to write the TB portion of the TB RAMs.  The address specified
                   ;         with the DEPOSIT command will be the actual RAM address accessed.  This
                   ;         routine will do any shifting necessary on the address specified to pos-
                   ;         ition it correctly.  The address specified has been loaded by the con-
                   ;         sole in LS location 5 prior to executing this routine.  The address
                   ;         must be in HEX format.
                   ;           The data specified has been placed in LS 6 by the console prior to
                   ;         executing this routine.  Bits 07-01 will be the Valid, Prot, Modify,
                   ;         and Byte Offset bits (there are four Prot bits).  Bits 22-08 will
                   ;         contain the PA bits from PA 23 through PA 09 respectively.  Bits 31
                   ;         through 23 and bit 0 are unused. This routine will do any shifting nec-
                   ;         essary to write the bits in TB as described.  The data specified must
                   ;         be in HEX format.
                   ;
                   ; CALLING SEQUENCE:
                   ;
                   ;         The console loads the address of a pointer to this routine (a JMP in-
                   ;         struction at WCS location 58) into the UPC and starts the CPU.
                   ;
                   ; INPUT PARAMETERS:
                   ;
                   ;         LS 5 - UNIBUS Map address in TB (must be in range 200-3FF HEX).
                   ;         LS 6 - Contains the data to write into the UNIBUS Map in bits 1-22.
                   ;
                   ; IMPLICIT INPUTS:
                   ;
                   ;         None
                   ;
                   ; OUTPUT PARAMATERS:
                   ;
                   ;         The UNIBUS map portion of the TB RAMs gets written with the data from
                   ;         LS 6 at the address specified in LS 5.
                   ;
                   ; IMPLICIT OUTPUTS:
                   ;
                   ;         None
                   ;
                   ; SIDE EFFECTS:
                   ;
                   ;         WR 0 will be modified
                   ;         WR 1 will be modified
                   ;
                   ;---

                   DEPOSIT.UBS:
U 0642, 08,64DC            JSR [SHIFT.UB.ADR]                    ;GO TO SUBROUTINE TO ARRANGE UNIBUS
```

```
                                                             ; MAP ADDRESS CORRECTLY
U 0643, 36,0C15          MOV LS[T6.SUB] TO WR[0]            ;GET DATA FROM LS 6
U 0644, 45,2615          BIC LS[#FF] TO WR[0]               ;CLEAR LOW BYTE OF DATA SPECIFIED.
                                                             ; LOW BYTE WILL BE PUT IN BITS 24-31
                                                             ; LATER
U 0645, 88,62AC          JSR [SHIFT.LOOP]                   ;GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
                                                             ; PLACES RIGHT
U 0646, E4,0C15          SWAP LS[T6.SUB] WITH WR[0]         ;SWAP ORIGINAL DATA SPECIFIED WITH MOD-
                                                             ; IFIED DATA.  NOW LS 6 CONTAINS BITS
                                                             ; 8-22 IN BITS 0-14 AS REQUIRED.  WR 0
                                                             ; CONTAINS ORIGINAL DATA SPECIFIED
U 0647, 45,2C15          BIC LS[#FFFFFF00] TO WR[0]         ;CLEAR ALL BUT LOW BYTE OF ORIGINAL
                                                             ; DATA SPECIFIED
U 0648, 88,62AC          JSR [SHIFT.LOOP]                   ;GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
                                                             ; PLACES RIGHT.  ON RETURN, WR 0 WILL
                                                             ; HAVE BITS 0-7 IN BITS 24-31, OTHER
                                                             ; BITS ARE CLEAR
U 0649, ED,0C15          BIS WR[0] TO LS[T6.SUB]            ;NOW LS 6 CONTAINS DATA TO WRITE IN
                                                             ; TB IN CORRECT FORMAT  I.E. PA BITS
                                                             ; IN BITS 00-14 AND BYTE OFFSET, MODIFY
                                                             ; PROT A THROUGH PROT D, AND VALID IN
                                                             ; BITS 25-31.

U 064A, 1B,0BF5          MEM.REQ[WRITE.UBS.MAP] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                                                             ; WRITE THE UNIBUS MAP
U 064B, B2,0C15          WRITE.MEM LS[T6.SUB]               ;WRITE DATA FROM LS 6 IN UNIBUS MAP
U 064C, 88,6874          JMP [WAIT.SUB]                     ;JUMP TO SET ATTN AND WAIT
                 ;
                 ;       SUBROUTINE TO POSITION ADDRESS SPECIFIED CORRECTLY TO ADDRESS THE
                 ;       UNIBUS MAP.
                 ;
                 SHIFT.UB.ADR:
U 064D, 36,0A15          MOV LS[T5.SUB] TO WR[0]            ;GET ADDRESS SPECIFIED IN WR 0
U 064E, A2,D015          SHL2 WR[0]                         ;SHIFT ADDRESS 2 PLACES LEFT
U 064F, A2,D015          SHL2 WR[0]                         ;SHIFT ADDRESS 2 PLACES LEFT
U 0650, A2,D015          SHL2 WR[0]                         ;SHIFT ADDRESS 2 PLACES LEFT
U 0651, A2,D015          SHL2 WR[0]                         ;SHIFT ADDRESS 2 PLACES LEFT
U 0652, 23,D015          ASHL WR[0]                         ;SHIFT LEFT ONE MORE PLACE
                                                             ; NOW ADDRESS IS IN BITS 9-17 OF WR 0
                         MOV WR[0] TO LS[T5.SUB],           ;NOW LS 5 CONTAINS CORRECT ADDRESS.
U 0653, 3E,0A14           RETURN                            ; RETURN TO MAIN
```

```
                 ;6165  .PAGE    "  EXAMINE UNIBUS MAP ROUTINE"
                 ;6166  ;+++
                 ;6167  ;        This routine will read the UNIBUS map portion of the memory controller.
                 ;6168  ;        The UNIBUS MAP contents will be returned with the byte offset, modify,
                 ;6169  ;        prot A through prot D, and valid in bits 01-07 respectively.  PA bits
                 ;6170  ;        09-23 are returned in bits 08-22 respectively.  Bits 23-31 and bit 0
                 ;6171  ;        are not used and so will be cleared before printing the result.  The
                 ;6172  ;        result will be put in LS 6 for the console to print.
                 ;6173  ;          The routine will issue a memory request with memory function=READ.UBS
                 ;6174  ;        .MAP and data type=LONGWORD.  The console will load LS 5 with the ad-
                 ;6175  ;        dress specified before executing this routine.  The routine will shift
                 ;6176  ;        the address around as required to address the UNIBUS map location spec-
                 ;6177  ;        ified.  The address specified must be in HEX format.
                 ;6178  ;          The UNIBUS map consists of 512 decimal locations (addresses from 200-
                 ;6179  ;        3FF)  The remaining locations in the TB RAM are either unused or used
                 ;6180  ;        by the translation buffer.  TB contents are read via the EXAMINE TB
                 ;6181  ;        routine.
                 ;6182  ;
                 ;6183  ; CALLING SEQUENCE:
                 ;6184  ;
                 ;6185  ;        The console loads the address of a pointer to this routine (a JMP in-
                 ;6186  ;        struction at WCS location 59) into the UPC and starts the CPU.
                 ;6187  ;
                 ;6188  ; INPUT PARAMETERS:
                 ;6189  ;
                 ;6190  ;        LS 5 - TB address in hex format
                 ;6191  ;
                 ;6192  ; IMPLICIT INPUTS:
                 ;6193  ;
                 ;6194  ;        None
                 ;6195  ;
                 ;6196  ; OUTPUT PARAMATERS:
                 ;6197  ;
                 ;6198  ;        LS 6 - UNIBUS map data from address specified
                 ;6199  ;
                 ;6200  ; IMPLICIT OUTPUTS:
                 ;6201  ;
                 ;6202  ;        None
                 ;6203  ;
                 ;6204  ; SIDE EFFECTS:
                 ;6205  ;
                 ;6206  ;        LS 5 is modified
                 ;6207  ;        WR 0 will be modified
                 ;6208  ;
                 ;6209  ;---
                 ;6210
                 ;6211  EXAMINE.UBS:
U 0654, 08,64DC  ;6212          JSR [SHIFT.UB.ADR]                  ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
                 ;6213                                              ; CORRECT POSITION AND REPLACE IN LS 5
U 0655, 1C,0B75  ;6214          MEM.REQ[READ.UBS.MAP] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                 ;6215                                              ; READ THE UBS MAP AT ADDRESS CONTAINED
                 ;6216                                              ; IN LS 5
U 0656, 30,2215  ;6217          MOV MEM.DATA TO WR[0]               ;PLACE RESULT IN WR 0
                 ;6218
U 0657, 45,2A15  ;6219          BIC LS[#FF000000] TO WR[0]          ;CLEAR UPPER BYTE OF RESULT (UPPER BYTE
                  ;6220                                          ; IS NOT PART OF TB AND IS UNPREDICT-
                  ;6221                                          ; ABLE)
U 0658, 45,6E15   ;6222          BIC LS[BIT23] TO WR[0]          ;DITTO FOR BIT 23
U 0659, 45,4015   ;6223          BIC LS[BIT0] TO WR[0]           ;DITTO FOR BIT 0.  WR 0 NOW CONTAINS
                  ;6224                                          ; RESULT TO PRINT
U 065A, BE,0C15   ;6225          MOV WR[0] TO LS[T6.SUB]         ;RESULT NOW IN LS 6
U 065B, 88,6874   ;6226          JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
                  ;6227                                          ; CONTROL
```

```
                    ;6228  .PAGE   "  DEPOSIT MAIN MEMORY ROUTINE"
                    ;6229  ;+++
                    ;6230  ;
                    ;6231  ; FUNCTIONAL DESCRIPTION:
                    ;6232  ;
                    ;6233  ;        This routine write a longword of data into main memory at the Physical
                    ;6234  ;        address specified.  This address need not be on a longword boundary.
                    ;6235  ;        The console will use this routine for data transfers to main memory as
                    ;6236  ;        well as for the DEPOSIT MM command.  The console will have loaded the
                    ;6237  ;        address into LS 5 and the data into LS 6 before executing this routine.
                    ;6238  ;          The routine issues a memory request with the memory function=WRITE.P
                    ;6239  ;        and the data type=longword.  It then issues a WRITE.MEM LS[6] instruc-
                    ;6240  ;        tion to write the data into memory.  A check on ERROR SUM is made to
                    ;6241  ;        assure that the write occured without an error.  If ERROR SUM is as-
                    ;6242  ;        serted, the CPU will issue a CPU ACK before asserting CPU ATTN to in-
                    ;6243  ;        form the console that an error occurred.
                    ;6244  ;
                    ;6245  ; CALLING SEQUENCE:
                    ;6246  ;
                    ;6247  ;        The console loads the address of a pointer to this routine (a JMP in-
                    ;6248  ;        struction at WCS location 54) into the UPC and starts the CPU.
                    ;6249  ;
                    ;6250  ; INPUT PARAMETERS:
                    ;6251  ;
                    ;6252  ;        LS 5 - Main memory physical address
                    ;6253  ;        LS 6 - Main memory longword data
                    ;6254  ;
                    ;6255  ; IMPLICIT INPUTS:
                    ;6256  ;
                    ;6257  ;        None
                    ;6258  ;
                    ;6259  ; OUTPUT PARAMATERS:
                    ;6260  ;
                    ;6261  ;        Main Memory address specified is written with data from LS 6.
                    ;6262  ;
                    ;6263  ; IMPLICIT OUTPUTS:
                    ;6264  ;
                    ;6265  ;        A memory error asserts CPU ACK to inform the console of a write error.
                    ;6266  ;
                    ;6267  ; SIDE EFFECTS:
                    ;6268  ;
                    ;6269  ;        None
                    ;6270  ;
                    ;6271  ;---
                    ;6272
                    ;6273  DEPOSIT.MM:
U 065C, 99,0A75     ;6274          MEM.REQ[WRITE.P] ADRS[T5.SUB] DT[LONG]  ;SET UP FOR WRITE TO MAIN
                    ;6275                                                  ; MEMORY USING THE PHYSICAL ADDRESS
                    ;6276                                                  ; STORED IN LS 5
                    ;6277          WRITE.MEM LS[T6.SUB],                   ;WRITE THE DATA FROM LS 6 INTO MAIN
                    ;6278                                                  ; MEMORY
U 065D, 32,0C1D     ;6279           SKIP.IF[MEM.REF.OK]                    ;SKIP NEXT INSTRUCTION IF NO MEMORY
                    ;6280                                                  ; ERROR (NO ERR SUM)
U 065E, 10,D015     ;6281          MISC [SET.CP.ACK]                       ;SET CPU ACK IF ERROR SUM ASSERTED
                    ;6282
```

```
U 065F, 36,4415   ;6283         MOV LS[#4] TO WR[0]               ;PUT A 4 IN WR 0
U 0660, 6C,0A15   ;6284         ADD WR[0] TO LS[T5.SUB]           ;INCREMENT LS 5 FOR NEXT LONGWORD DEP
U 0661, 88,6874   ;6285         JMP [WAIT.SUB]                    ;JUMP TO SET ATTN AND WAIT
```

```
                  ;6286  .PAGE   " EXAMINE MAIN MEMORY ROUTINE"
                  ;6287  ;+++
                  ;6288  ;
                  ;6289  ; FUNCTIONAL DESCRIPTION:
                  ;6290  ;
                  ;6291  ;       This routine is used to read a longword form main memory at the Phy-
                  ;6292  ;       sical address specified.  The address need not be on a longword bound-
                  ;6293  ;       ary.  This routine is used by the console on an EXAMINE.MM command.
                  ;6294  ;       The console will have loaded LS 5 with the physical address specified
                  ;6295  ;       before executing this routine.
                  ;6296  ;         The routine will first write CSR 1 to set the INH REP CRD (inhibit
                  ;6297  ;       report correctable read data) bit to prevent an error sum from occuring
                  ;6298  ;       on a single bit error which has been corrected.  Then the routine is-
                  ;6299  ;       sues a memory request with memory function=READ.P and DT=longword.  It
                  ;6300  ;       then reads the memory location and puts the result in LS 6.  It also
                  ;6301  ;       checks for an ERROR SUM and issues a CPU ACK if an error occurred (sin-
                  ;6302  ;       gle bit errors that have been corrected will not cause an ERR SUM).  If
                  ;6303  ;       an error occurred, CPU ACK will be set before issueing CPU ATTN to in-
                  ;6304  ;       form the console that an error occurred.
                  ;6305  ;
                  ;6306  ; CALLING SEQUENCE:
                  ;6307  ;
                  ;6308  ;       The console loads the address of a pointer to this routine (a JMP in-
                  ;6309  ;       struction at WCS location 55) into the UPC and starts the CPU.
                  ;6310  ;
                  ;6311  ; INPUT PARAMETERS:
                  ;6312  ;
                  ;6313  ;       LS 5 - Physical address in main memory to read.
                  ;6314  ;
                  ;6315  ; IMPLICIT INPUTS:
                  ;6316  ;
                  ;6317  ;       LS[INH.CRD] - Data to write into CSR 1 to inhibit error sum on a CRD.
                  ;6318  ;       LS[CSR1]  - Address for writing CSR 1.
                  ;6319  ;
                  ;6320  ; OUTPUT PARAMATERS:
                  ;6321  ;
                  ;6322  ;       LS 6 - Longword data from physical memory address specified.
                  ;6323  ;
                  ;6324  ; IMPLICIT OUTPUTS:
                  ;6325  ;
                  ;6326  ;       CPU ACK if an error sum occurs.
                  ;6327  ;
                  ;6328  ; SIDE EFFECTS:
                  ;6329  ;
                  ;6330  ;       None
                  ;6331  ;
                  ;6332  ;---
                  ;6333
                  ;6334  EXAMINE.MM:
U 0662, 1D,45F5   ;6335          MEM.REQ[WRITE.CSR] ADRS[CSR1] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                  ;6336                                                  ; WRITE CSR 1
U 0663, B2,7815   ;6337          WRITE.MEM LS[INH.CRD]                   ;SET INH REP CRD IN CSR 1 AND CLEAR
                  ;6338                                                  ; ECC DISABLE
                  ;6339
U 0664, 18,0BF5   ;6340          MEM.REQ[READ.P] ADRS[T5.SUB] DT[LONG]   ;ISSUE MEMORY REQUEST TO
```

```
                  ;6341                                    ; READ LONGWORD DATA FROM MAIN MEMORY
                  ;6342                                    ; PHYSICAL ADDRESS IN LS 5
                  ;6343         MOV MEM.DATA TO LS[T6.SUB], ;READ DATA AND PUT IN LS 6
U 0665, 38,0C1D   ;6344          SKIP.IF[MEM.REF.OK]       ; SKIP NEXT INSTRUCTION IF NO MEMORY
                  ;6345                                    ; ERROR (NO ERROR SUM)
U 0666, 10,D015   ;6346         MISC [SET.CP.ACK]          ;SET CPU ACK IF AN ERROR OCCURRED
U 0667, 88,6874   ;6347         JMP [WAIT.SUB]             ;JUMP TO SET ATTN AND WAIT
```

```
                   ;6348  .page    "  EXAMINE IDC ROUTINES"
                   ;6349  ;+++
                   ;6350  ;
                   ;6351  ; FUNCTIONAL DESCRIPTION:
                   ;6352  ;
                   ;6353  ;        The following routines are used to read data from the optional IDC
                   ;6354  ;        module's registers.  The result is put in LS 6 for the console to
                   ;6355  ;        print out.
                   ;6356  ;
                   ;6357  ; CALLING SEQUENCE:
                   ;6358  ;
                   ;6359  ;        The console loads the address of a pointer to this routine (a JMP in-
                   ;6360  ;        struction at WCS location 5A through 5E) into the UPC and starts the
                   ;6361  ;        CPU.
                   ;6362  ;
                   ;6363  ; INPUT PARAMETERS:
                   ;6364  ;
                   ;6365  ;        None
                   ;6366  ;
                   ;6367  ; IMPLICIT INPUTS:
                   ;6368  ;
                   ;6369  ;        None
                   ;6370  ;
                   ;6371  ; OUTPUT PARAMATERS:
                   ;6372  ;
                   ;6373  ;        LS 6 - Data from IDC register specified.
                   ;6374  ;
                   ;6375  ; IMPLICIT OUTPUTS:
                   ;6376  ;
                   ;6377  ;        None
                   ;6378  ;
                   ;6379  ; SIDE EFFECTS:
                   ;6380  ;
                   ;6381  ;        None
                   ;6382  ;
                   ;6383  ;---
                   ;6384
                   ;6385  EXAMINE.ICSR:
U 0668, 90,9015    ;6386          MISC [SET.PORT.I.O]               ;SELECT PORT TO BUS
U 0669, 97,8015    ;6387          PORT.READ PORT.ADRS[CSR]          ;SEND THE READ COMMAND
U 066A, 08,6764    ;6388          JMP [READ.IDC]                    ;READ THE DATA
                   ;6389
                   ;6390  EXAMINE.IDAR:
U 066B, 90,9015    ;6391          MISC [SET.PORT.I.O]               ;SELECT PORT TO BUS
U 066C, 17,8415    ;6392          PORT.READ PORT.ADRS[DAR]          ;SEND THE READ COMMAND
U 066D, 08,6764    ;6393          JMP [READ.IDC]                    ;READ THE DATA
                   ;6394
                   ;6395  EXAMINE.DBUF:
U 066E, 90,9015    ;6396          MISC [SET.PORT.I.O]               ;SELECT PORT TO BUS
U 066F, 97,8C15    ;6397          PORT.READ PORT.ADRS[DATA.WORD]    ;SEND THE READ COMMAND
U 0670, 08,6764    ;6398          JMP [READ.IDC]                    ;READ THE DATA
                   ;6399
                   ;6400  EXAMINE.PATT:
U 0671, 90,9015    ;6401          MISC [SET.PORT.I.O]               ;SELECT PORT TO BUS
U 0672, 17,9015    ;6402          PORT.READ PORT.ADRS[PATTERN]      ;SEND THE READ COMMAND
```

```
U 0673, 08,6764   ;6403           JMP [READ.IDC]                    ;READ THE DATA
                  ;6404
                  ;6405   EXAMINE.POSIT:
U 0674, 90,9015   ;6406           MISC [SET.PORT.I.O]               ;SELECT PORT TO BUS
U 0675, 97,9415   ;6407           PORT.READ PORT.ADRS[POSITION]     ;SEND THE READ COMMAND
                  ;6408
                  ;6409
                  ;6410   READ.IDC:
U 0676, 3A,0C15   ;6411           MOV PORT TO LS[T6.SUB]            ;READ DATA AND PUT IN LS 6
U 0677, 10,8015   ;6412           MISC [SET.ACC.I.O]                ;RETURN SELECT TO ACCELERATOR
U 0678, 88,6874   ;6413           JMP [WAIT.SUB]                    ;JUMP TO SET ATTN AND WAIT
```

```
                     ;6414  .page   "  DEPOSIT IDC ROUTINES"
                     ;6415  ;+++
                     ;6416  ;
                     ;6417  ; FUNCTIONAL DESCRIPTION:
                     ;6418  ;
                     ;6419  ;       The following routines are used to write data to the optional IDC
                     ;6420  ;       module's registers.  The data is taken from LS 6 which is previously
                     ;6421  ;       loaded by the monitor when the DEPOSIT command is executed.
                     ;6422  ;
                     ;6423  ; CALLING SEQUENCE:
                     ;6424  ;
                     ;6425  ;       The console loads the address of a pointer to this routine (a JMP in-
                     ;6426  ;       struction at WCS location 5F through 61) into the UPC and starts the
                     ;6427  ;       CPU.
                     ;6428  ;
                     ;6429  ; INPUT PARAMETERS:
                     ;6430  ;
                     ;6431  ;       LS 6 - Data pattern to write.
                     ;6432  ;
                     ;6433  ; IMPLICIT INPUTS:
                     ;6434  ;
                     ;6435  ;       None
                     ;6436  ;
                     ;6437  ; OUTPUT PARAMATERS:
                     ;6438  ;
                     ;6439  ;       None
                     ;6440  ;
                     ;6441  ; IMPLICIT OUTPUTS:
                     ;6442  ;
                     ;6443  ;       None
                     ;6444  ;
                     ;6445  ; SIDE EFFECTS:
                     ;6446  ;
                     ;6447  ;       None
                     ;6448  ;
                     ;6449  ;---
                     ;6450
                     ;6451  DEPOSIT.ICSR:
U 0679, 36,0C15      ;6452          MOV LS[T6.SUB] TO WR[0]          ;GET DATA PATTERN TO WRITE
U 067A, 17,A015      ;6453          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;WRITE TO IDC
U 067B, 88,6874      ;6454          JMP [WAIT.SUB]                   ;JUMP TO SET ATTN AND WAIT
                     ;6455
                     ;6456
                     ;6457  DEPOSIT.IDAR:
U 067C, 36,0C15      ;6458          MOV LS[T6.SUB] TO WR[0]          ;GET DATA PATTERN TO WRITE
U 067D, 97,A415      ;6459          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ;WRITE TO IDC
U 067E, 88,6874      ;6460          JMP [WAIT.SUB]                   ;JUMP TO SET ATTN AND WAIT
                     ;6461
                     ;6462
                     ;6463  DEPOSIT.DBUF:
U 067F, 36,0C15      ;6464          MOV LS[T6.SUB] TO WR[0]          ;GET DATA PATTERN TO WRITE
U 0680, 17,AC15      ;6465          PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0]     ;WRITE TO IDC
U 0681, 88,6874      ;6466          JMP [WAIT.SUB]                   ;JUMP TO SET ATTN AND WAIT
                     ;6467
                     ;6468
```

```
                  ;6469  CLEAR.FIFO.ADDR:
U 0682, 17,C015   ;6470          PORT.CONTROL [CLEAR.FIFO.CNTR]  ;CLEAR ADDRESS IN FIFO A OR FIFO B
U 0683, 88,6874   ;6471          JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
                  ;6472
                  ;6473
                  ;6474  SELECT.FIFO.A:
U 0684, 17,D815   ;6475          PORT.CONTROL [SEL.FIFO.A]       ;SELECT FIFO A
U 0685, 88,6874   ;6476          JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
                  ;6477
                  ;6478
                  ;6479  SELECT.FIFO.B:
U 0686, 97,DC15   ;6480          PORT.CONTROL [SEL.FIFO.B]       ;SELECT FIFO B
                  ;6481
                  ;6482  WAIT.SUB:
U 0687, 10,E015   ;6483          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                  ;6484  WAIT.DE.EX:
U 0688, 88,6884   ;6485          JMP [WAIT.DE.EX]                ;LOOP SELF UNTIL CONSOLE TAKES CONTROL
```

```
                    ;6486  .PAGE    "  SAVE WR ROUTINE"
                    ;6487  ;+++
                    ;6488  ;
                    ;6489  ; FUNCTIONAL DESCRIPTION:
                    ;6490  ;
                    ;6491  ;       This routine saves WRs 0-3 in LS locations 1-4.  When the console takes
                    ;6492  ;       control of the CPU, it calls this routine so that the working registers
                    ;6493  ;       can be used by the console.  The WRs are restored by another routine
                    ;6494  ;       called by the console before the CPU resumes execution (on a COntinue
                    ;6495  ;       command, for example).
                    ;6496  ;         This routine simply moves data from WR 0-3 to LS locations 1-4 and
                    ;6497  ;       then issues a CPU ATTN to the console.  It then loops on a JMP self
                    ;6498  ;       until the console takes over.
                    ;6499  ;
                    ;6500  ; CALLING SEQUENCE:
                    ;6501  ;
                    ;6502  ;       The console loads the address of a pointer to this routine (a JMP in-
                    ;6503  ;       struction at WCS location 46) into the UPC and starts the CPU.
                    ;6504  ;
                    ;6505  ; INPUT PARAMETERS:
                    ;6506  ;
                    ;6507  ;       None
                    ;6508  ;
                    ;6509  ; IMPLICIT INPUTS:
                    ;6510  ;
                    ;6511  ;       None
                    ;6512  ;
                    ;6513  ; OUTPUT PARAMATERS:
                    ;6514  ;
                    ;6515  ;       LS 1 - Contents of WR 0
                    ;6516  ;       LS 2 - Contents of WR 1
                    ;6517  ;       LS 3 - Contents of WR 2
                    ;6518  ;       LS 4 - Contents of WR 3
                    ;6519  ;
                    ;6520  ; IMPLICIT OUTPUTS:
                    ;6521  ;
                    ;6522  ;       None
                    ;6523  ;
                    ;6524  ; SIDE EFFECTS:
                    ;6525  ;
                    ;6526  ;       None
                    ;6527  ;
                    ;6528  ;---
                    ;6529
                    ;6530  SAVE.WR:
                    ;6531
U 0689, 3E,0215     ;6532          MOV WR[0] TO LS[SAV.WR0]        ;SAVE WR 0 IN LS LOCATION 1
U 068A, BE,0495     ;6533          MOV WR[1] TO LS[SAV.WR1]        ;SAVE WR 1 IN LS LOCATION 2
U 068B, 3E,0715     ;6534          MOV WR[2] TO LS[SAV.WR2]        ;SAVE WR 2 IN LS LOCATION 3
U 068C, 3E,0995     ;6535          MOV WR[3] TO LS[SAV.WR3]        ;SAVE WR 3 IN LS LOCATION 4
U 068D, 88,6874     ;6536          JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
```

```
                 .PAGE    "  RESTORE WR ROUTINE"
                 ;+++
                 ;
                 ; FUNCTIONAL DESCRIPTION:
                 ;
                 ;       This routine restores the 2901 working registers save in LS 1-4.  The
                 ;       console calls this routine before the CPU is restarted after it has
                 ;       been halted by the console.
                 ;         The routine simply moves the data from LS locations 1-4 to WRs 0-3
                 ;       and then issues a CPU ATTN to the console.  A JMP self is executed next
                 ;       waiting for the console to take control.
                 ;
                 ; CALLING SEQUENCE:
                 ;
                 ;       The console loads the address of a pointer to this routine (a JMP in-
                 ;       struction at WCS location 47) into the UPC and starts the CPU.
                 ;
                 ; INPUT PARAMETERS:
                 ;
                 ;       LS 1 contains the data to be restored in WR 0.
                 ;       LS 2 contains the data to be restored in WR 1.
                 ;       LS 3 contains the data to be restored in WR 2.
                 ;       LS 4 contains the data to be restored in WR 3.
                 ;
                 ; IMPLICIT INPUTS:
                 ;
                 ;       None
                 ;
                 ; OUTPUT PARAMATERS:
                 ;
                 ;       WR 0 gets data from LS 1.
                 ;       WR 1 gets data from LS 2.
                 ;       WR 2 gets data from LS 3.
                 ;       WR 3 gets data from LS 4.
                 ;
                 ; IMPLICIT OUTPUTS:
                 ;
                 ;       None
                 ;
                 ; SIDE EFFECTS:
                 ;
                 ;       None
                 ;
                 ;---

                 RESTORE.WR:
U 068E, B6,0215          MOV LS[SAV.WR0] TO WR[0]        ;RESTORE WR 0
U 068F, 36,0495          MOV LS[SAV.WR1] TO WR[1]        ;RESTORE WR 1
U 0690, B6,0715          MOV LS[SAV.WR2] TO WR[2]        ;RESTORE WR 2
U 0691, B6,0995          MOV LS[SAV.WR3] TO WR[3]        ;RESTORE WR 3
U 0692, 88,6874          JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
```

```
;6588  .PAGE   "WCS SUBROUTINES FOR CPU USE"
;6589  .TOC    "  ERROR ROUTINE"
;6590  ;+++
;6591  ;
;6592  ; FUNCTIONAL DESCRIPTION:
;6593  ;
;6594  ;       This routine is used to detect and report diagnostic errors occurring
;6595  ;       during WCS based micro-diagnostics.  The WCS micro-code sets up the
;6596  ;       parameters listed below and calls this routine.  The routine compares
;6597  ;       WR 1 (expected data) and WR 0 (received data) according to the error
;6598  ;       mask.  Bits set in the mask indicate bits that are not checked for
;6599  ;       errors.
;6600  ;        If WR 0 and WR 1 are equal (no error) the routine executes a return+1
;6601  ;       to the calling point.  The only exception to this is if loop on error
;6602  ;       is set.  In that case, the error number is checked to see if that error
;6603  ;       occurred previously.  If so, then the error loop is forced by doing a
;6604  ;       return.  This will lock an error loop in on intermittent errors.
;6605  ;        If WR 0 and WR 1 are not equal (an error) then bit 8 is set in the
;6606  ;       CONTROL AND STATUS word, expected and received data is set up in LS for
;6607  ;       printout, and flags are checked in the ERROR CONTROL word.  This con-
;6608  ;       trols whether printouts are disabled, loop on error is enabled, etc.
;6609  ;       After printing the error, a return+1 is made to the calling point.
;6610  ;        Loop on error causes a return rather than return+1 to cause an error
;6611  ;       loop.  Also CPU ACKNOWLEDGE is toggled at the end of the routine for
;6612  ;       a scope sync point.
;6613  ;
;6614  ; CALLING SEQUENCE:
;6615  ;
;6616  ;       JSR [CHECK.RESULT]
;6617  ;
;6618  ; INPUT PARAMETERS:
;6619  ;
;6620  ;       WR[0] = Received data i.e. the actual result of the test
;6621  ;       WR[1] = Expected data i.e. what the result should have been
;6622  ;
;6623  ; IMPLICIT INPUTS:
;6624  ;
;6625  ;       LS[ERROR.MASK] = Used to mask out bits that are not to be tested
;6626  ;       LS[ERROR.NUMBER] = Current error number
;6627  ;       LS[PREVIOUS.ERROR] = Error number of previous data
;6628  ;       LS[CONTROL.STATUS] = Control and Status word.  Contains information
;6629  ;           written by the CPU to inform the monitor what to do.
;6630  ;       LS[ERROR.CONTROL] = Information such as loop-on-error, bell-on-error,
;6631  ;           no error report, and halt on error.
;6632  ;
;6633  ; OUTPUT PARAMATERS:
;6634  ;
;6635  ;       NONE
;6636  ;
;6637  ; IMPLICIT OUTPUTS:
;6638  ;
;6639  ;       LS[CONTROL.STATUS] = Control and Status with new status information
;6640  ;       LS[PREVIOUS.ERROR] = Last error number (modified only if error and
;6641  ;           loop on error is enabled)
;6642  ;       LS[EXPECTED.DATA] = Expected result if an error was detected
```

```
                    ;6643   ;       LS[RECEIVED.DATA] = Actual result if an error was detected
                    ;6644   ;
                    ;6645   ; SIDE EFFECTS:
                    ;6646   ;
                    ;6647   ;       WR[0], WR[1], and the Q reg are modified and are not restored before
                    ;6648   ;           returning.
                    ;6649   ;       If loop-on-error is enabled and an error loop is to be executed, CPU
                    ;6650   ;           ACKNOWLEDGE will be toggled for a scope sync.
                    ;6651   ;
                    ;6652   ;---
                    ;6653
                    ;6654   CHECK.RESULT:
U 0693, 45,8A15     ;6655           BIC LS[ERROR.MASK] TO WR[0]     ;MASK OFF NOT TESTED BITS (CLEAR THEM)
                    ;6656                                           ; FOR RECEIVED DATA
U 0694, C5,8A95     ;6657           BIC LS[ERROR.MASK] TO WR[1]     ;DITTO FOR EXPECTED DATA
                    ;6658
                    ;6659           XOR WR[0] WITH WR[1] TO Q,      ;CMP RECEIVED DATA IN WR[0] WITH
U 0695, AB,80B5     ;6660            DT(LONG)&SET.ALU.CC            ; EXPECTED DATA IN WR[1] FOR ERROR
U 0696, 88,69F1     ;6661           JMP.IF[NEQ] TO [ERROR]          ;JUMP IF ERROR
                    ;6662
U 0697, 36,9015     ;6663           MOV LS[ERROR.CONTROL] TO WR[0]  ;GET ERROR CONTROL WORD FROM LS
                    ;6664           BIT WR[0] WITH LS[LOE],         ;SEE IF LOOP ON ERROR IS ENABLED
U 0698, 59,4035     ;6665            DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 0699, DB,0001     ;6666           SKIP.IF[BIT.SET]                ;SKIP IF IT IS
U 069A, DB,0016     ;6667           RETURN+1                        ;ELSE RETURN+1 (NO ERROR AND NO LOOP)
                    ;6668
U 069B, 36,A015     ;6669           MOV LS[PREVIOUS.ERROR] TO WR[0] ;IF NO ERROR BUT LOOP ON ERROR IS
                    ;6670                                           ; ENABLED, SEE IF THERE WAS A
                    ;6671                                           ; PREVIOUS ERROR
                    ;6672           XOR WR[0] WITH LS[ERROR.NUMBER] TO Q,
U 069C, CD,8235     ;6673            DT(LONG)&SET.ALU.CC            ;IS THIS THE SAME SPOT THAT HAD AN
                    ;6674                                           ; ERROR BEFORE? (INTERMITTANT ERROR)
U 069D, 08,6B91     ;6675           JMP.IF[NEQ] TO [RET+1]          ;JUMP IF NOT TO RETURN WITHOUT ERROR
                    ;6676                                           ; LOOPING (RETURN+1)
U 069E, 08,6BC4     ;6677           JMP [RET]                       ;ELSE CONTINUE TO LOOP (WE WILL LOOP
                    ;6678                                           ; ON ERROR EVEN IF IT GOES AWAY)
                    ;6679
                    ;6680   ERROR:
U 069F, 3E,8615     ;6681           MOV WR[0] TO LS[RECEIVED.DATA]  ;PUT RESULT OF TEST IN LS FOR PRINTOUT
U 06A0, 36,9015     ;6682           MOV LS[ERROR.CONTROL] TO WR[0]  ;GET ERROR CONROL BITS TO CHECK FOR
                    ;6683                                           ; LOOP COMMAND
                    ;6684           BIT WR[0] WITH LS[LOOP.COMMAND],;SEE IF BIT 6 SET
U 06A1, 59,4C35     ;6685            DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 06A2, 08,6BC1     ;6686           JMP.IF[BIT.SET] TO [RET]        ;GO LOOP IF SET (FAST LOOP)
                    ;6687
U 06A3, 3E,8495     ;6688           MOV WR[1] TO LS[EXPECTED.DATA]  ;PUT EXPECTED DATA IN LS FOR PRINTOUT
                    ;6689
U 06A4, 36,8095     ;6690           MOV LS[CONTROL.STATUS] TO WR[1] ;GET CONTROL AND STATUS WORD FROM LS
U 06A5, C5,3295     ;6691           BIC LS[#FE7FFFFF] TO WR[1]      ;CLEAR ALL BUT BITS 5&23&24 IN CONTROL
                    ;6692                                           ; AND STATUS WORD
U 06A6, C7,5095     ;6693           BIS LS[ERROR] TO WR[1]          ;SET BIT 8 IN CONTROL AND STATUS WORD
                    ;6694                                           ; (INDICATES AN ERROR WAS DETECTED)
U 06A7, BE,8095     ;6695           MOV WR[1] TO LS[CONTROL.STATUS] ;WRITE UPDATED CONTROL AND STATUS WORD
                    ;6696                                           ; BACK TO LS[40]
                    ;6697
```

```
                   ;6698            BIT WR[0] WITH LS[BELL],           ;SEE IF BELL ON ERROR IS SET (WR 0 STILL
                   ;6699                                               ; CONTAINS ERROR CONTROL WORD)
U 06A8, D9,4435    ;6700             DT(LONG)&SET.ALU.CC               ; SET CONDITION CODES
U 06A9, 88,6AD9    ;6701            JMP.IF[BITS.CLR] TO [CHECK.NER]    ;IF BELL ON ERROR IS NOT SET, JUMP TO
                   ;6702                                               ; SEE IF ERROR REPORTING IS INHIBITED
U 06AA, 10,E015    ;6703            MISC [SET.CP.ATTN]                 ;ELSE SET CPU ATTN (RING MY BELL)
                   ;6704
U 06AB, 08,6AB4    ;6705   WAIT.1:  JMP [WAIT.1]                       ;WAIT FOR CONSOLE TO STOP ME
U 06AC, 08,6B64    ;6706            JMP [LOOP]                         ;GO TO CHECK LOOP-ON-ERROR AFTER
                   ;6707                                               ; CONSOLE HAS FINISHED
                   ;6708
                   ;6709   CHECK.NER:
                   ;6710            BIT WR[0] WITH LS[NER],            ;IF NO BELL SEE IF ERROR REPORT IS
                   ;6711                                               ; INHIBITED
U 06AD, D9,4235    ;6712             DT(LONG)&SET.ALU.CC               ; SET CONDITION CODES
U 06AE, 88,6B21    ;6713            JMP.IF[BIT.SET] TO [CHECK.HALT]    ;JUMP IF ERROR REPORT IS INHIBITED TO
                   ;6714                                               ; CHECK FOR HALT ON ERROR
                   ;6715
U 06AF, 10,E015    ;6716            MISC [SET.CP.ATTN]                 ;SET CPU ATTN (REPORT ERROR)
U 06B0, 08,6B04    ;6717   WAIT.2:  JMP [WAIT.2]                       ;WAIT FOR CONSOLE TO STOP ME
U 06B1, 08,6B64    ;6718            JMP [LOOP]                         ;GO TO CHECK LOOP-ON-ERROR AFTER
                   ;6719                                               ; CONSOLE HAS FINISHED
                   ;6720
                   ;6721   CHECK.HALT:
                   ;6722            BIT WR[0] WITH LS[HOE],            ;SEE IF HALT ON ERROR IS ENABLED
U 06B2, 59,4635    ;6723             DT(LONG)&SET.ALU.CC               ; SET CONDITION CODES
U 06B3, 88,6B69    ;6724            JMP.IF[BITS.CLR] TO [LOOP]         ;JUMP IF NOT TO CHECK LOOP-ON-ERROR
                   ;6725
U 06B4, 10,E015    ;6726            MISC [SET.CP.ATTN]                 ;ELSE SET CPU ATTN (HALT ON ERROR)
U 06B5, 08,6B54    ;6727   WAIT.3:  JMP [WAIT.3]                       ;WAIT FOR CONSOLE TO STOP ME
                   ;6728
U 06B6, 36,9015    ;6729   LOOP:    MOV LS[ERROR.CONTROL] TO WR[0]     ;GET ERROR CONTROL BITS (NER, HOE ETC.)
                   ;6730            BIT WR[0] WITH LS[LOE],            ;SEE IF LOOP-ON-ERROR
U 06B7, 59,4035    ;6731             DT(LONG)&SET.ALU.CC               ; SET CONDITION CODES
U 06B8, DB,0001    ;6732            SKIP.IF[BIT.SET]                   ; SKIP IF LOOP ON ERROR IS ENABLED
                   ;6733
U 06B9, DB,0016    ;6734   RET+1:   RETURN+1                           ;ELSE RETURN+1 FOR NO ERROR LOOP
                   ;6735
U 06BA, 36,8215    ;6736            MOV LS[ERROR.NUMBER] TO WR[0]      ;GET ERROR NUMBER IF LOOPING
                   ;6737
U 06BB, BE,A015    ;6738            MOV WR[0] TO LS[PREVIOUS.ERROR]    ;AND SAVE IT IN PREVIOUS ERROR
                   ;6739
U 06BC, 10,D015    ;6740   RET:     MISC [SET.CP.ACK]                  ;SET CPU ACKNOWLEDGE FOR SCOPE SYNC
                   ;6741            MISC [CLR.CP.ATTN.AND.ACK],        ;AND THEN CLEAR IT
U 06BD, 10,C014    ;6742             RETURN                            ; RETURN TO TEST AND LOOP ON ERROR
                   ;6743
```

```
                    ;6744   .PAGE
                    ;6745   ;
                    ;6746   ;
                    ;6747   ;
                    ;6748   ERR.NO.TEST:
U 06BE, DB,0015     ;6749           NOP                             ;NOP WILL CAUSE A SEQUENCE ERROR
U 06BF, B6,5E15     ;6750           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                    ;6751                                           ; STATUS WORD
U 06C0, 3E,8015     ;6752           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                    ;6753                                           ; BIT 15 INDICATES START OF TEST TO
                    ;6754                                           ; CONSOLE
U 06C1, 88,6874     ;6755           JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
                    ;6756   .TOC    "START TRANSFER ROUTINE"
                    ;6757   ;+++
                    ;6758   ;       This routine loads the LS with information neccesary for these tests
                    ;6759   ;       to run.  It will then issue a CPU ATTN with the control and status word
                    ;6760   ;       clear to indicate that all transfers are complete.
                    ;6761   ;
                    ;6762   ;       RESULT: LS LOADED WITH CONSTANTS. MONITOR INFORMED TRANSFERS OVER.
                    ;6763   ;---
                    ;6764
                    ;6765   TRANSFER.POINT:
                    ;6766
                    ;6767
U 06C2, 2F,8015     ;6768           CLR WR[0]                       ;START WITH 0 IN WR[0]
U 06C3, BF,FE15     ;6769           MOV WR[0] TO LS[PSL.HW]         ;MAKE SURE COMPAT MODE IS CLEAR
                    ;6770
U 06C4, 20,4015     ;6771           INC WR[0]                       ;NOW HAVE 1 IN WR[0]
U 06C5, 23,D015     ;6772           ASHL WR[0]                      ;10(B)
U 06C6, 20,4015     ;6773           INC WR[0]                       ;11(B)
U 06C7, 23,D015     ;6774           ASHL WR[0]                      ;110(B)
U 06C8, 20,4015     ;6775           INC WR[0]                       ;111(B)
U 06C9, 23,D015     ;6776           ASHL WR[0]                      ;1110(B)
U 06CA, 23,D015     ;6777           ASHL WR[0]                      ;1 1100(B)
U 06CB, 23,D015     ;6778           ASHL WR[0]                      ;11 1000(B)
U 06CC, 23,D015     ;6779           ASHL WR[0]                      ;111 0000(B)  NOW HAVE 70(H) IN WR 0.
                    ;6780                                           ; THIS IS CORRECT START ADDRESS OF DATA
                    ;6781                                           ; SECTION (INCLUDING LONGWORD COUNT,
                    ;6782                                           ; DESTINATION, AND DESTINATION ADDRESS)
U 06CD, 3E,0E15     ;6783           MOV WR[0] TO LS[TRANSFER.POINTER]  ;LOAD LS 7 WITH POINTER TO TRANSFER
                    ;6784                                           ; DATA SECTION.
                    ;6785
U 06CE, 2F,8015     ;6786           CLR WR[0]                       ;0 IN WR 0
U 06CF, 20,4015     ;6787           INC WR[0]                       ;1 IN WR 0
U 06D0, 23,D015     ;6788           ASHL WR[0]                      ;10(B)
U 06D1, 23,D015     ;6789           ASHL WR[0]                      ;100(B)
U 06D2, 23,D015     ;6790           ASHL WR[0]                      ;1000(B)
U 06D3, 23,D015     ;6791           ASHL WR[0]                      ;1 0000(B)
U 06D4, 23,D015     ;6792           ASHL WR[0]                      ;10 0000(B)
U 06D5, 23,D015     ;6793           ASHL WR[0]                      ;100 0000(B)
U 06D6, 23,D015     ;6794           ASHL WR[0]                      ;1000 0000(B)
U 06D7, 23,D015     ;6795           ASHL WR[0]                      ;1 0000 0000(B)
U 06D8, 23,D015     ;6796           ASHL WR[0]                      ;10 0000 0000(B)
U 06D9, 23,D015     ;6797           ASHL WR[0]                      ;100 0000 0000(B)
U 06DA, 23,D015     ;6798           ASHL WR[0]                      ;1000 0000 0000(B)
```

```
U 06DB, 23,D015   ;6799           ASHL WR[0]                          ;1 0000 0000 0000(B)
U 06DC, 23,D015   ;6800           ASHL WR[0]                          ;10 0000 0000 0000(B)
U 06DD, 3E,8015   ;6801           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 13 IN LS CONTROL AND STATUS
                  ;6802                                               ; WORD (BIT 13 INDICATES CONSOLE MUST
                  ;6803                                               ; PERFORM A DATA TRANSFER)
U 06DE, 10,E015   ;6804           MISC [SET.CP.ATTN]                  ;ISSUE CPU ATTN TO CONSOLE (DO FIRST LS
                  ;6805                                               ; TRANSFER TO LOCATIONS 0 THROUGH 77(H))
                  ;6806   WAIT.XFER1:
U 06DF, 08,6DF4   ;6807           JMP [WAIT.XFER1]                    ;LOOP HERE WAITING FOR CONSOLE TO RE-
                  ;6808                                               ; SPOND
                  ;6809
U 06E0, 36,CA15   ;6810           MOV LS[#162(H)] TO WR[0]            ;GET START ADDRESS OF SECOND HALF OF LS
                  ;6811                                               ; TRANSFER
U 06E1, 3E,0E15   ;6812           MOV WR[0] TO LS[TRANSFER.POINTER] ;LOAD START ADDRESS IN LS FOR CONSOLE
U 06E2, 36,5A15   ;6813           MOV LS[XFER.DATA] TO WR[0]          ;BIT 13 INDICATES TO DO DATA TRANSFER
U 06E3, 3E,8015   ;6814           MOV WR[0] TO LS[CONTROL.STATUS] ;SET UP CONTROL AND STATUS WORD
U 06E4, 10,E015   ;6815           MISC [SET.CP.ATTN]                  ;ISSUE CPU ATTN TO CONSOLE (DO SECOND LS
                  ;6816                                               ; TRANSFER TO LOCATIONS 80 THROUGH F7(H))
                  ;6817   WAIT.XFER2:
U 06E5, 08,6E54   ;6818           JMP [WAIT.XFER2]                    ;LOOP HERE WAITING FOR CONSOLE TO RE-
                  ;6819                                               ; SPOND
                  ;6820
U 06E6, 9B,9D75   ;6821           MEM.REQ[READ.V.RCHK.IFILL] ADRS[ZERO] DT[LONG]
                  ;6822                                               ;THIS INSTRUCTION USED TO FREE MEMORY
                  ;6823                                               ; IF HUNG WAITING FOR A DATA RCVD.
U 06E7, 30,2215   ;6824           MOV MEM.DATA TO WR[0]               ;THIS INSTRUCTION USED TO FREE MEMORY
                  ;6825                                               ; IF HUNG IN A OCTA FLOW
U 06E8, 30,2215   ;6826           MOV MEM.DATA TO WR[0]               ;NEEDS 4 OF THESE
U 06E9, 30,2215   ;6827           MOV MEM.DATA TO WR[0]
U 06EA, 30,2215   ;6828           MOV MEM.DATA TO WR[0]               ;DONE WITH MEMORY INITIALIZATION
                  ;6829
U 06EB, 17,CC15   ;6830           PORT.CONTROL [CLEAR.IDC]            ;INITIALIZE IDC TO IDLE LOOP
U 06EC, 17,CC15   ;6831           PORT.CONTROL [CLEAR.IDC]
U 06ED, 17,CC15   ;6832           PORT.CONTROL [CLEAR.IDC]
                  ;6833
U 06EE, 17,D415   ;6834           PORT.CONTROL [CLEAR.AUTO]           ;CLEAR AUTOMODE
                  ;6835
U 06EF, 36,7815   ;6836           MOV LS[BIT28] TO WR[0]
U 06F0, B6,4E95   ;6837           MOV LS[BIT7] TO WR[1]
U 06F1, AE,C215   ;6838           BIS WR[1] TO WR[0]
U 06F2, 17,A015   ;6839           PORT.WRITE PORT.ADRS[CSR] WITH WR[0]   ;SET WRITE INHIBIT IN IDC CSR
                  ;6840                                                  ; AND SET CRDY
                  ;6841
U 06F3, 36,6015   ;6842           MOV LS[INTERUPT.EN] TO WR[0]        ;SET BIT 16 IN WR
U 06F4, 3E,8015   ;6843           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 16 TO CLEAR ANY INTERRUPTS
                  ;6844                                               ; AND INFORM CONSOLE THAT XFERS ARE
                  ;6845                                               ; DONE
U 06F5, 10,E015   ;6846           MISC [SET.CP.ATTN]                  ;CALL CONSOLE
                  ;6847   WAIT.XFER:
U 06F6, 88,6F64   ;6848           JMP [WAIT.XFER]                     ;WAIT FOR CONSOLE TO RESPOND
U 06F7, 5B,0014   ;6849           RETURN                              ;USED WHEN TRANSFER IS CALLED FROM
                  ;6850                                               ; MICRO-DIAGNOSTIC ONLY
                  ;6851
```

```
;6852   .PAGE
;6853   .REGION/700,3FFF
;6854
```

```
                ;6855  .PAGE    "SUBROUTINES FOR IDC MICRODIAGNOSTIC"
                ;6856  .TOC     " WAIT FOR CRDY TO SET ROUTINE (NON DATA TRANSFER)"
                ;6857  ;+++
                ;6858  ;
                ;6859  ; FUNCTIONAL DESCRIPTION:
                ;6860  ;
                ;6861  ;       This subroutine will check the CRDY bit of CSR for set after an issuance
                ;6862  ;       of a non-data-block transfer command between the disk and the controller.
                ;6863  ;       The CSR is read and the subroutine will keep looping in itself until
                ;6864  ;       bit <7> is found to be set.
                ;6865  ;
                ;6866  ; CALLING SEQUENCE:
                ;6867  ;
                ;6868  ;       JSR [WAIT.CSR.CRDY]
                ;6869  ;
                ;6870  ; INPUT PARAMETERS:
                ;6871  ;
                ;6872  ;       NONE
                ;6873  ;
                ;6874  ; IMPLICIT INPUTS:
                ;6875  ;
                ;6876  ;       NONE
                ;6877  ;
                ;6878  ; OUTPUT PARAMATERS:
                ;6879  ;
                ;6880  ;       NONE
                ;6881  ;
                ;6882  ; IMPLICIT OUTPUTS:
                ;6883  ;
                ;6884  ;       NONE
                ;6885  ;
                ;6886  ; SIDE EFFECTS:
                ;6887  ;
                ;6888  ;       LS[T5.SUB] will be modified
                ;6889  ;
                ;6890  ; RETURNS:
                ;6891  ;
                ;6892  ;       RETURN - normal return to main program body
                ;6893  ;       RETURN+1 - invalid return indicating CRDY is not set
                ;6894  ;---
                ;6895
                ;6896  WAIT.CSR.CRDY:
U 0700, 90,9015 ;6897          MISC [SET.PORT.I.O]                     ;select from port to bus
U 0701, 97,8015 ;6898          PORT.READ PORT.ADRS[CSR]                ;read the csr
U 0702, 3A,0A15 ;6899          MOV PORT TO LS[T5.SUB]                  ;save the csr word
U 0703, 36,4E15 ;6900          MOV LS[SETCSR.CRDY] TO WR[0]            ;get the crdy mask bit
                ;6901          AND LS[T5.SUB] TO WR[0],                ;is crdy set ?
U 0704, C2,0A35 ;6902           DT(LONG)&SET.ALU.CC
U 0705, 5B,0009 ;6903          SKIP.IF[EQL]                            ;no
U 0706, 5B,0014 ;6904          RETURN                                  ;yes, valid return
U 0707, DB,0016 ;6905          RETURN+1                                ;wait loop
                ;6906
```

```
                  ;6907  .PAGE
                  ;6908  .TOC     " WAIT FOR DRDY TO SET ROUTINE (NON DATA TRANSFER)"
                  ;6909  ;+++
                  ;6910  ;
                  ;6911  ; FUNCTIONAL DESCRIPTION:
                  ;6912  ;
                  ;6913  ;        This subroutine will check the DRDY bit of CSR for set after an issuance
                  ;6914  ;        of a non-data-block transfer command between the disk and the controller.
                  ;6915  ;        The CSR is read and the subroutine will keep looping in itself until
                  ;6916  ;        bit <0> is found to be set.
                  ;6917  ;
                  ;6918  ; CALLING SEQUENCE:
                  ;6919  ;
                  ;6920  ;        JSR [WAIT.CSR.DRDY]
                  ;6921  ;
                  ;6922  ; INPUT PARAMETERS:
                  ;6923  ;
                  ;6924  ;        NONE
                  ;6925  ;
                  ;6926  ; IMPLICIT INPUTS:
                  ;6927  ;
                  ;6928  ;        NONE
                  ;6929  ;
                  ;6930  ; OUTPUT PARAMATERS:
                  ;6931  ;
                  ;6932  ;        NONE
                  ;6933  ;
                  ;6934  ; IMPLICIT OUTPUTS:
                  ;6935  ;
                  ;6936  ;        NONE
                  ;6937  ;
                  ;6938  ; SIDE EFFECTS:
                  ;6939  ;
                  ;6940  ;        LS[PORTO] will be modified
                  ;6941  ;
                  ;6942  ; RETURN:
                  ;6943  ;
                  ;6944  ;        RETURN - normal return to main program body
                  ;6945  ;---
                  ;6946
                  ;6947  WAIT.CSR.DRDY:
U 0708, 90,9015   ;6948         MISC [SET.PORT.I.O]                          ;select from port to bus
U 0709, 97,8015   ;6949         PORT.READ PORT.ADRS[CSR]                     ;read the CSR back
U 070A, 38,3215   ;6950         MOV PORT TO LS[PORTO]                        ; from the port and
U 070B, 37,3215   ;6951         MOV LS[PORTO] TO WR[0]                       ;  load it in WRO
                  ;6952         BIT LS[BITO] WITH WR[0],                     ;check if DRDY is set
U 070C, 59,4035   ;6953          DT(LONG)&SET.ALU.CC
U 070D, 08,70F1   ;6954         JMP.IF[BIT.SET] TO [DRDY.1]                  ;exit to main program
U 070E, 88,7084   ;6955         JMP [WAIT.CSR.DRDY]                          ;wait some more
                  ;6956
U 070F, 58,0014   ;6957  DRDY.1: RETURN
                  ;6958
```

```
                    ;6959  .PAGE
                    ;6960  .TOC     "GET A CHARACTER FROM THE DATA BUFFER ROUTINE"
                    ;6961  ;+++
                    ;6962  ;
                    ;6963  ; FUNCTIONAL DESCRIPTION:
                    ;6964  ;
                    ;6965  ;        This subroutine will read a data byte from the currently selected FIFO
                    ;6966  ;        as a longword whose upper bytes will be masked out, it will then
                    ;6967  ;        initialize the shift counter for data bits used in the "LONG WORD LABEL"
                    ;6968  ;        routine. In fact this routine is intended to be used exclusively by
                    ;6969  ;        the "LONG WORD LABEL" routine.
                    ;6970  ;
                    ;6971  ; CALLING SEQUENCE:
                    ;6972  ;
                    ;6973  ;        JSR [GET.A.CHAR]
                    ;6974  ;
                    ;6975  ; INPUT PARAMETERS:
                    ;6976  ;
                    ;6977  ;        NONE
                    ;6978  ;
                    ;6979  ; IMPLICIT INPUTS:
                    ;6980  ;
                    ;6981  ;        Label data in FIFO A
                    ;6982  ;
                    ;6983  ; OUTPUT PARAMATERS:
                    ;6984  ;
                    ;6985  ;        WR[0] - one ASCII character in hex value in its lowest byte
                    ;6986  ;
                    ;6987  ; IMPLICIT OUTPUTS:
                    ;6988  ;
                    ;6989  ;        NONE
                    ;6990  ;
                    ;6991  ; SIDE EFFECTS:
                    ;6992  ;
                    ;6993  ;        LS[T5.SUB] will be modified
                    ;6994  ;
                    ;6995  ; RETURNS:
                    ;6996  ;
                    ;6997  ;        RETURN - normal return to main program body
                    ;6998  ;
                    ;6999  ;---
                    ;7000
                    ;7001  GET.A.CHAR:                                     ;get the a char.
U 0710, 17,8815     ;7002          PORT.READ PORT.ADRS[DATA.BYTE]          ; of the label from the disk
U 0711, 3A,0A15     ;7003          MOV PORT TO LS[T5.SUB]
U 0712, 36,0A15     ;7004          MOV LS[T5.SUB] TO WR[0]                 ; and store it up
U 0713, C2,2615     ;7005          AND LS[#FF] TO WR[0]                    ;mask out upper bytes
U 0714, 2F,8295     ;7006          CLR WR[1]                               ;init. shift counter
U 0715, 5B,0014     ;7007          RETURN
                    ;7008
                    ;7009  .PAGE
                    ;7010  .TOC     "STORE A LABEL IN A LONGWORD ROUTINE"
                    ;7011  ;+++
                    ;7012  ;
                    ;7013  ; FUNCTIONAL DESCRIPTION:
                    ;7014  ;
                    ;7015  ;       This subroutine will call the "GET.A.CHAR" each time when it fetches a
                    ;7016  ;       character. It will then shift the received character byte into the
                    ;7017  ;       right format of a label. This is done four times so as to fill up an
                    ;7018  ;       entire longword with ASCII characters in hex value.
                    ;7019  ;
                    ;7020  ; CALLING SEQUENCE:
                    ;7021  ;
                    ;7022  ;       JSR [LONG.WORD.LABEL]
                    ;7023  ;
                    ;7024  ; INPUT PARAMETERS:
                    ;7025  ;
                    ;7026  ;       NONE
                    ;7027  ;
                    ;7028  ; IMPLICIT INPUTS:
                    ;7029  ;
                    ;7030  ;       NONE
                    ;7031  ;
                    ;7032  ; OUTPUT PARAMATERS:
                    ;7033  ;
                    ;7034  ;       LS[SAV.WRO] - 4 ASCII characters in hex value
                    ;7035  ;
                    ;7036  ; IMPLICIT OUTPUTS:
                    ;7037  ;
                    ;7038  ;       NONE
                    ;7039  ;
                    ;7040  ; SIDE EFFECTS:
                    ;7041  ;
                    ;7042  ;       LS[SAV.WRO] will be modified
                    ;7043  ;
                    ;7044  ; RETURNS:
                    ;7045  ;
                    ;7046  ;       RETURN - normal return to main program body
                    ;7047  ;
                    ;7048  ;---
                    ;7049
                    ;7050  LONG.WORD.LABEL:                                    ;get the 1st char. and
U 0716, 08,710C     ;7051          JSR [GET.A.CHAR]                            ; shift that in place
                    ;7052
                    ;7053  RO.R.8BIT:
U 0717, 23,4015     ;7054          ROR WR[0]                                   ;rotate right by 1 bit
U 0718, 20,4295     ;7055          INC WR[1]                                   ;inc. counter
                    ;7056          CMP LS[#8] WITH WR[1],                      ;enough shift ?
U 0719, CE,46B5     ;7057           DT(LONG)&SET.ALU.CC
U 071A, 08,7171     ;7058          JMP.IF[NEQ] TO [RO.R.8BIT]                  ;no, do another shift
U 071B, 3E,0215     ;7059          MOV WR[0] TO LS[SAV.WRO]                    ;yes, save it
                    ;7060
U 071C, 08,710C     ;7061          JSR [GET.A.CHAR]                            ;get the 1st char. and
                    ;7062                                                      ; shift that in place
                    ;7063  RO.R.16BIT:
```

```
U 071D, 23,4015  ;7064           ROR WR[0]                          ;rotate right by 1 bit
U 071E, 20,4295  ;7065           INC WR[1]                          ;inc. counter
                 ;7066           CMP LS[#10] WITH WR[1],            ;enough shift ?
U 071F, 4E,48B5  ;7067            DT(LONG)&SET.ALU.CC
U 0720, 08,71D1  ;7068           JMP.IF[NEQ] TO [RO.R.16BIT]        ;no, do another shift
U 0721, C7,0215  ;7069           BIS LS[SAV.WRO] TO WR[0]           ;combine the 2 char.
U 0722, 3E,0215  ;7070           MOV WR[0] TO LS[SAV.WRO]           ;save this
                 ;7071
                 ;7072
U 0723, 08,710C  ;7073           JSR [GET.A.CHAR]                   ;get the 1st char. and
                 ;7074                                              ; shift that in place
                 ;7075  RO.L.8BIT:
U 0724, A3,C015  ;7076           ROL WR[0]                          ;rotate left by 1 bit
U 0725, 20,4295  ;7077           INC WR[1]                          ;inc. counter
                 ;7078           CMP LS[#8] WITH WR[1],             ;enough shift ?
U 0726, CE,46B5  ;7079            DT(LONG)&SET.ALU.CC
U 0727, 08,7241  ;7080           JMP.IF[NEQ] TO [RO.L.8BIT]         ;no, do another shift
U 0728, C7,0215  ;7081           BIS LS[SAV.WRO] TO WR[0]           ;combine the 3 char.
U 0729, 3E,0215  ;7082           MOV WR[0] TO LS[SAV.WRO]           ;save this
                 ;7083
                 ;7084  RO.ZERO.BIT:                                ;the char. already in place
U 072A, 08,710C  ;7085           JSR [GET.A.CHAR]
U 072B, C7,0215  ;7086           BIS LS[SAV.WRO] TO WR[0]           ;combine the 4 char. to form
U 072C, 3E,0215  ;7087           MOV WR[0] TO LS[SAV.WRO]           ; a long word and save it
                 ;7088
U 072D, 5B,0014  ;7089           RETURN
                 ;7090
                 ;7091
                 ;7092
```

```
                .PAGE
                .TOC    "SHIFT RL02 HEADER ROUTINE"
                ;+++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This routine is used to read the 2 RL02 header bytes from FIFO A and
                ;       arrange them into the DAR bit format.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       JSR [SHIFT.HDR]
                ;
                ; INPUT PARAMETERS:
                ;
                ;       NONE
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       RL02 header in FIFO A
                ;
                ; OUTPUT PARAMATERS:
                ;
                ;       LS[SAV.WR0] - DAR cylinder difference format
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; SIDE EFFECTS:
                ;
                ;       LS[PORT0], LS[PORT1] & LS[SAV.WR0] will be modified
                ;
                ; RETURNS:
                ;
                ;       RETURN - normal return to main program body
                ;
                ;---

                SHIFT.HDR:
U 072E, 17,D815         PORT.CONTROL [SEL.FIFO.A]               ;select FIFO A
U 072F, 17,C015         PORT.CONTROL [CLEAR.FIFO.CNTR]          ;clear the FIFO A adr cntr
U 0730, 90,9015         MISC [SET.PORT.I.O]                     ;select the IDC
U 0731, 17,8815         PORT.READ PORT.ADRS[DATA.BYTE]          ;initiate IDC read
U 0732, 38,3215         MOV PORT TO LS[PORT0]                   ;load low byte
U 0733, 17,8815         PORT.READ PORT.ADRS[DATA.BYTE]          ;initiate another idc read
U 0734, 38,3415         MOV PORT TO LS[PORT1]                   ;load high byte
U 0735, 36,2695         MOV LS[#FF] TO WR[1]                    ;mask for data
U 0736, 37,3215         MOV LS[PORT0] TO WR[0]
U 0737, 2F,C215         AND WR[1] TO WR[0]                      ;mask off low byte
U 0738, BF,3215         MOV WR[0] TO LS[PORT0]
U 0739, 37,3415         MOV LS[PORT1] TO WR[0]
U 073A, 2F,C215         AND WR[1] TO WR[0]                      ;mask off high byte
U 073B, A2,D015         SHL2 WR[0]                              ;shift byte up 2 bits
U 073C, A2,D015         SHL2 WR[0]                              ;shift byte up 2 bits
```

```
U 073D, A2,D015   ;7148          SHL2 WR[0]                                  ;shift byte up 2 bits
U 073E, A2,D015   ;7149          SHL2 WR[0]                                  ;shift byte up 2 bits
U 073F, B7,3295   ;7150          MOV LS[PORT0] TO WR[1]                      ;retrieve low byte
U 0740, AE,C215   ;7151          BIS WR[1] TO WR[0]                          ;header in rite dar format
U 0741, B7,EC95   ;7152          MOV LS[#0000FF80] TO WR[1]                  ;mask for current cylinder
U 0742, 2F,C215   ;7153          AND WR[1] TO WR[0]                          ; address
U 0743, 3E,0215   ;7154          MOV WR[0] TO LS[SAV.WR0]                    ;save the shifted header
U 0744, 5B,0014   ;7155          RETURN
                  ;7156
                  ;7157
```

```
                        ;7158   .PAGE
                        ;7159   .TOC     "DAR FORMAT FOR SEEK ROUTINE"
                        ;7160   ;+++
                        ;7161   ;
                        ;7162   ; FUNCTIONAL DESCRIPTION:
                        ;7163   ;
                        ;7164   ;       The routine is used to fix up the DAR word with the applicable cylinder
                        ;7165   ;       difference, marker bit and seek higher or lower address direction bit.
                        ;7166   ;       The output of this routine is the DAR word ready for a seek command
                        ;7167   ;       issued to the RL02 disk drive.
                        ;7168   ;
                        ;7169   ; CALLING SEQUENCE:
                        ;7170   ;
                        ;7171   ;       JSR [SEEK.DAR]
                        ;7172   ;
                        ;7173   ; INPUT PARAMETERS:
                        ;7174   ;
                        ;7175   ;       LS[SAV.WRO] - cylinder difference in DAR format from JSR [SHIFT.HDR]
                        ;7176   ;       LS[SEEK.CYL] - the cylinder address to be sought
                        ;7177   ;
                        ;7178   ; IMPLICIT INPUTS:
                        ;7179   ;
                        ;7180   ;       NONE
                        ;7181   ;
                        ;7182   ; OUTPUT PARAMATERS:
                        ;7183   ;
                        ;7184   ;       LS[SAV.WRO] - complete DAR format for RL02 seek command
                        ;7185   ;
                        ;7186   ; IMPLICIT OUTPUTS:
                        ;7187   ;
                        ;7188   ;       NONE
                        ;7189   ;
                        ;7190   ; SIDE EFFECTS:
                        ;7191   ;
                        ;7192   ;       LS[SAV.WRO] will be modified
                        ;7193   ;
                        ;7194   ; RETURNS:
                        ;7195   ;
                        ;7196   ;       RETURN - normal return to main program body
                        ;7197   ;
                        ;7198   ;---
                        ;7199
                        ;7200   SEEK.DAR:
U 0745, B6,0215         ;7201           MOV LS[SAV.WRO] TO WR[0]
U 0746, 37,8495         ;7202           MOV LS[SEEK.CYL] TO WR[1]                       ;fetch the seek cyl. adr.
                        ;7203           SUB WR[1] FROM WR[0],                           ;get the cylinder difference
U 0747, AE,4235         ;7204            DT(LONG)&SET.ALU.CC
U 0748, DB,001C         ;7205           SKIP.IF[LSS]
U 0749, 88,74C4         ;7206           JMP [SK.DOWN]
                        ;7207   SK.UP:
U 074A, 20,8015         ;7208           NEG WR[0]                                       ;form positive difference
U 074B, 47,4415         ;7209           BIS LS[BIT2] TO WR[0]                           ;set direction bit
                        ;7210   SK.DOWN:
U 074C, C7,4015         ;7211           BIS LS[BIT0] TO WR[0]                           ;set marker bit in either case
U 074D, 3E,0215         ;7212           MOV WR[0] TO LS[SAV.WRO]                        ;save up the dar format
```

```
U 074E, 58,0014  ;7213          RETURN
                 ;7214
```

```
                        .PAGE
                        .TOC    "RL02 SEEK AND VERIFY ROUTINE"
                        ;+++
                        ;
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       This routine is used to seek to a predetermined disk cylinder input,
                        ;       and then verify the location of the heads by doing a read header
                        ;       function, reading the header from FIFO A, calling JSR [SHIFT.HDR]
                        ;       to arrange the cylinder address in DAR format and setting up error
                        ;       parameters to be handled by the main program body.
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       JSR [SEEK.VERIFY]
                        ;
                        ; INPUT PARAMETERS:
                        ;
                        ;       LS[SAV.WR0] - cylinder difference in RL02 DAR format from JSR [SHIFT.HDR]
                        ;
                        ; IMPLICIT INPUTS:
                        ;
                        ;       NONE
                        ;
                        ; OUTPUT PARAMATERS:
                        ;
                        ;       LS[SAV.WR0] - complete cylinder address in RL02 DAR format for JSR [CHECK.RESULT]
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;
                        ;       Error reporting parameters for JSR [CHECK.RESULT]
                        ;
                        ; SIDE EFFECTS:
                        ;
                        ;       LS[SAV.WR0] will be modified
                        ;
                        ; RETURNS:
                        ;
                        ;       RETURN - normal return to main program body
                        ;
                        ;---

                        SEEK.VERIFY:
U 074F, B6,0215                 MOV LS[SAV.WR0] TO WR[0]                ;retrieve the preset DAR
U 0750, 97,A415                 PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ;write DAR with this format

U 0751, B7,0495                 MOV LS[SEEK] TO WR[1]                   ;get the seek command
U 0752, C7,5095                 BIS LS[SETCSR.DS1] TO WR[1]             ;select drive 1
U 0753, 97,A095                 PORT.WRITE PORT.ADRS[CSR] WITH WR[1]    ;seek to the next cyl.

                                ;************************************************************
                                ;* Refer to the RL02 SEEK IDC function codes as listed above *
                                ;*                        in test 5                          *
                                ;************************************************************
```

```
                  ;7270  WAIT.SK.VER.1:
U 0754, 88,700C   ;7271          JSR [WAIT.CSR.CRDY]                        ;check if CRDY is set
U 0755, 5B,001E   ;7272          SKIP.IF[SKIP]                              ;CRDY set
U 0756, 88,7544   ;7273          JMP [WAIT.SK.VER.1]                        ;wait some more
                  ;7274
                  ;7275  ;Wait for DRDY to set to make sure the drive is ready to take the next command :
                  ;7276
U 0757, 08,708C   ;7277          JSR [WAIT.CSR.DRDY]                        ;wait for DRDY to set
                  ;7278
                  ;7279  ;Read the header and find out if the head is really on the innermost cylinder.
                  ;7280
                  ;7281  VERIFY.1:
U 0758, B7,0615   ;7282          MOV LS[READ.HEADER] TO WR[0]               ;get the read_header command
U 0759, 47,5015   ;7283          BIS LS[SETCSR.DS1] TO WR[0]                ;select drive 1
U 075A, 17,A015   ;7284          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]       ;write the CSR
                  ;7285                                                     ;MAINT=0,CRDY=0,F<2:0>=100
                  ;7286
                  ;7287          ;*****************************************************************
                  ;7288          ;* Refer to the RL02 read header idc function codes as listed above *
                  ;7289          ;*                        in test 1                              *
                  ;7290          ;*****************************************************************
                  ;7291
                  ;7292  WAIT.SK.VER.2:
U 075B, 88,700C   ;7293          JSR [WAIT.CSR.CRDY]                        ;check if CRDY is set
U 075C, 88,75E4   ;7294          JMP [VERIFY.2]                             ;yes, CRDY is set now
U 075D, 88,75B4   ;7295          JMP [WAIT.SK.VER.2]                        ;no, wait some more
                  ;7296
                  ;7297  VERIFY.2:
U 075E, 88,72EC   ;7298          JSR [SHIFT.HDR]                            ;shift the header to DAR format
                  ;7299
U 075F, B7,A295   ;7300          MOV LS[#FFFF007F] TO WR[1]                 ;error mask used
U 0760, BE,8A95   ;7301          MOV WR[1] TO LS[ERROR.MASK]
                  ;7302
U 0761, 5B,0014   ;7303          RETURN
                  ;7304
```

```
                    .PAGE  "Test 1  Y BUS VERIFICATION TEST (M8388 IDC MODULE OR M8390 CPU MODULE)"
                    ;++                                   (Not run in CRD mode)
                    ;
                    ; Test Description:
                    ;
                    ;       This test will insure that the Y bus is fully operational as seen by
                    ;       the CPU. The test will move data from local store to a working
                    ;       register and then use the Y Bus to move the data back to a temporary
                    ;       location in local store.
                    ;       The data patterns that will be used are:
                    ;               1. AAAAAAAA
                    ;               2. 55555555
                    ;               3. 33333333
                    ;               4. 0F0F0F0F
                    ;               5. 00FF00FF
                    ;               6. 0000FFFF
                    ;
                    ;
                    ; Assumptions:
                    ;
                    ;       It is assumed that the CPU has been tested and was found to
                    ;       be fully operational without an IDC module installed.
                    ;
                    ; Test Steps:
                    ;
                    ;       1. Set up error mask, error number,and module number in LS
                    ;          (for error printouts) and clear previous error number in LS.
                    ;       2. MOV data pattern from LS to WR[1].
                    ;       3. MOV WR[1] to a temporary LS location.
                    ;       4. MOV the temporary LS location to WR[0] and check.
                    ;       5. Repeat test steps 1-4 until all data patterns have been tested.
                    ;
                    ; Error:
                    ;
                    ;       ERROR 1 - Y Bus failed.
                    ;
                    ;                 Printout:
                    ;                         EXPECTED           RECEIVED
                    ;                          WR[0]              WR[0]
                    ; Debug:
                    ;
                    ;  ERROR 1:
                    ;
                    ;       If the CPU Micro Diagnostic was run with no errors prior to the
                    ;       addition of the IDC module to the system, then the Y Bus is most
                    ;       likely being corrupted by the IDC module.
                    ;
                    ;       If the CPU Micro Diagnostics were not run prior to beginning
                    ;       this test then either the CPU or the FPA Module (if one exists) or
                    ;       the IDC module could be at fault.
                    ;
                    ;--

                    T.1:
U 0762, B6,5E15             MOV LS[BEGIN.TEST] TO WR[0]           ;SET BIT 15 IN WR[0] FOR
```

```
                                                                ; CONTROL AND STATUS WORD
U 0763, 3E,8015              MOV WR[0] TO LS[CONTROL.STATUS]    ;SET BIT 15 IN CONTROL AND
                                                                ; STATUS WORD -  BIT 15
                                                                ; INDICATES START OF TEST TO
                                                                ; CONSOLE
U 0764, 10,E015              MISC [SET.CP.ATTN]                 ;ISSUE CPU ATTN TO CONSOLE
                     WAIT.T1.0:
U 0765, 08,7654              JMP [WAIT.T1.0]                    ;LOOP HERE WAITING FOR CONSOLE
                                                                ; TO RESPOND
U 0766, 65,A015              CLR LS[PREVIOUS.ERROR]             ;CLEAR PREVIOUS ERROR NUMBER
                                                                ; IN LS

U 0767, 36,4215              MOV LS[CPU] TO WR[0]               ;STORE MODULE CODE IN LS TO
U 0768, C7,4A15              BIS LS[IDC] TO WR[0]               ; INDICATE IDC AND CPU
U 0769, 3E,8C15              MOV WR[0] TO LS[MODULE.NUM]        ; MODULES

U 076A, B6,4015              MOV LS[#1] TO WR[0]                ;SET WR0 = 1
U 076B, BE,8215              MOV WR[0] TO LS[ERROR.NUMBER]      ;SET UP ERROR NUMBER = 1 IN LS

U 076C, E5,8A15              CLR LS[ERROR.MASK]                 ;SET UP ERROR MASK TO CHECK
                                                                ; ALL BITS

                     LOOP.T1.1:
U 076D, B6,9815              MOV LS[#AAAAAAAA] TO WR[0]         ;LOAD THE WORKING REGISTER
U 076E, BE,1415              MOV WR[0] TO LS[T10]               ;MOV THE DATA OVER THE Y BUS
U 076F, 36,1415              MOV LS[T10] TO WR[0]               ;GET THE RECEIVED DATA
U 0770, 36,9895              MOV LS[#AAAAAAAA] TO WR[1]         ;GET THE EXPECTED DATA
U 0771, 08,693C              JSR [CHECK.RESULT]                 ;CHECK THE RESULT
U 0772, 88,76D4              JMP [LOOP.T1.1]                    ;ERROR LOOP
                     LOOP.T1.2:
U 0773, 36,9A15              MOV LS[#55555555] TO WR[0]
U 0774, BE,1415              MOV WR[0] TO LS[T10]               ;MOV THE DATA OVER THE Y BUS
U 0775, 36,1415              MOV LS[T10] TO WR[0]               ;GET THE RECEIVED DATA
U 0776, B6,9A95              MOV LS[#55555555] TO WR[1]         ;GET THE EXPECTED DATA
U 0777, 08,693C              JSR [CHECK.RESULT]                 ;CHECK THE RESULT
U 0778, 88,7734              JMP [LOOP.T1.2]                    ;ERROR LOOP
                     LOOP.T1.3:
U 0779, B6,C415              MOV LS[#33333333] TO WR[0]
U 077A, BE,1415              MOV WR[0] TO LS[T10]               ;MOV THE DATA OVER THE Y BUS
U 077B, 36,1415              MOV LS[T10] TO WR[0]               ;GET THE RECEIVED DATA
U 077C, 36,C495              MOV LS[#33333333] TO WR[1]         ;GET THE EXPECTED DATA
U 077D, 08,693C              JSR [CHECK.RESULT]                 ;CHECK THE RESULT
U 077E, 88,7794              JMP [LOOP.T1.3]                    ;ERROR LOOP
                     LOOP.T1.4:
U 077F, 36,C615              MOV LS[#0F0F0F0F] TO WR[0]
U 0780, BE,1415              MOV WR[0] TO LS[T10]               ;MOV THE DATA OVER THE Y BUS
U 0781, 36,1415              MOV LS[T10] TO WR[0]               ;GET THE RECEIVED DATA
U 0782, B6,C695              MOV LS[#0F0F0F0F] TO WR[1]         ;GET THE EXPECTED DATA
U 0783, 08,693C              JSR [CHECK.RESULT]                 ;CHECK THE RESULT
U 0784, 88,77F4              JMP [LOOP.T1.4]                    ;ERROR LOOP
                     LOOP.T1.5:
U 0785, B6,C815              MOV LS[#00FF00FF] TO WR[0]
U 0786, BE,1415              MOV WR[0] TO LS[T10]               ;MOV THE DATA OVER THE Y BUS
U 0787, 36,1415              MOV LS[T10] TO WR[0]               ;GET THE RECEIVED DATA
U 0788, 36,C895              MOV LS[#00FF00FF] TO WR[1]         ;GET THE EXPECTED DATA
```

```
U 0789, 08,693C  ;7415          JSR [CHECK.RESULT]                      ;CHECK THE RESULT
U 078A, 88,7854  ;7416          JMP [LOOP.T1.5]                         ;ERROR LOOP
                 ;7417  LOOP.T1.6:
U 078B, B7,A615  ;7418          MOV LS[#0000FFFF] TO WR[0]
U 078C, BE,1415  ;7419          MOV WR[0] TO LS[T10]                    ;MOV THE DATA OVER THE Y BUS
U 078D, 36,1415  ;7420          MOV LS[T10] TO WR[0]                    ;GET THE RECEIVED DATA
U 078E, 37,A695  ;7421          MOV LS[#0000FFFF] TO WR[1]              ;GET THE EXPECTED DATA
U 078F, 08,693C  ;7422          JSR [CHECK.RESULT]                      ;CHECK THE RESULT
U 0790, 08,78B4  ;7423          JMP [LOOP.T1.6]                         ;ERROR LOOP
                 ;7424  END.T1:                                         ;END TEST
                 ;7425
                 ;7426
                 ;7427
```

```
.PAGE  "Test 2  IDC SIZING TEST (M8388 IDC MODULE OR M8390 CPU MODULE)"
;++                             (not run in CRD mode)
;
; Test Description:
;
;           If this diagnostic is being run under APT and an IDC is not found
;           then an error will be reported.
;           This test will size for the IDC Module by writing and then reading
;           Bit <0> of the Disk Address Register (DAR). If this cannot be
;           accomplished successfully, then the message "IDC NOT PRESENT" will
;           be printed and the IDC microdiagnostic will not be executed. Otherwise
;           "IDC PRESENT" will be printed and test execution will continue.
;
; Assumptions:
;
;           It is assumed that the CPU and the Y bus are fully operational.
;
; Test Steps:
;
;           1. Move zero to WR[0].
;           2. Move #1 to WR [1].
;           3. Write a one to DAR <0>.
;           4. Move WR[0] to a temporary LS location to change the data on the
;              Y BUS.
;           5. Read DAR <0>.
;           6. Check result. If incorrect data, print message "IDC NOT PRESENT" and
;              go to Step 12.
;           7. Write a zero to DAR <0>.
;           8. Move WR[1] to a temporary LS location to change the data on the
;              Y BUS.
;           9. Read DAR <0>.
;          10. Check result.
;          11. If data is incorrect, print message "IDC NOT PRESENT".
;          12. If APT is present then force an error because IDC was not found.
;          13. Go to end of section.
;          14. Else print "IDC PRESENT" and continue IDC microdiagnostic execution.
;
; Errors:
;
;  ERROR 1 - The diagnostic was being run under APT and no IDC was found.
;
;                    Printout:
;                                EXPECTED       RECEIVED
;                                  N/A            N/A
;
; Debug:
;
;           If an IDC is not found by this test and one is known to exist, then
;           TEST 4 DISK ADDRESS REGISTER TEST should be run to allow looping on
;           error. A more detailed debug section is also available in TEST 4.
;
;           If the IDC is not found and one is known to exist then the fault could
;           be in one of the following:
;
;           1. The STATE REG, MISC DECODER PAL on the CPU module which generates
```

```
                    ;7483  ;         the PORT INSTR H signal.
                    ;7484  ;      2. The A,B ADRS GEN PAL on the CPU module which generates the
                    ;7485  ;         READ PORT L signal.
                    ;7486  ;      3. Bit <0> of the data bus transceiver
                    ;7487  ;      4. Bit <0> of the IDC I/O bus
                    ;7488  ;      5. Bit <0> of the DAR
                    ;7489  ;      6. The DAR control signals, WRITE DAR or SEL DAR which are decoded from
                    ;7490  ;         the Port instruction through the PORT INTERFACE REG PAL, the PORT
                    ;7491  ;         RD/WR DECODE PAL and the Port instruction decoder.
                    ;7492  ;
                    ;7493  ;--
                    ;7494  ;*****************************************************************************
                    ;7495  T.2:
U 0791, B6,5E15     ;7496          MOV LS[BEGIN.TEST] TO WR[0]              ;SET BIT 15 IN WR[0] FOR
                    ;7497                                                   ; CONTROL AND STATUS WORD
U 0792, 3E,8015     ;7498          MOV WR[0] TO LS[CONTROL.STATUS]          ;SET BIT 15 IN CONTROL AND
                    ;7499                                                   ; STATUS WORD - BIT 15
                    ;7500                                                   ; INDICATES START OF TEST TO
                    ;7501                                                   ; CONSOLE
U 0793, 10,E015     ;7502          MISC [SET.CP.ATTN]                       ;ISSUE CPU ATTN TO CONSOLE
                    ;7503  WAIT.T2.0:
U 0794, 88,7944     ;7504          JMP [WAIT.T2.0]                          ;LOOP HERE WAITING FOR CONSOLE
                    ;7505                                                   ; TO RESPOND
                    ;7506
U 0795, E5,0E15     ;7507          CLR LS[T7.SUB]                           ;SET UP FOR MESSAGE PRINTOUT
U 0796, B6,4015     ;7508          MOV LS[#1] TO WR[0]
U 0797, BE,8215     ;7509          MOV WR[0] TO LS[ERROR.NUMBER]            ;SET ERROR NUMBER =1
                    ;7510
U 0798, E5,8A15     ;7511          CLR LS[ERROR.MASK]                       ;SET MASK TO CHECK ALL BITS
                    ;7512
                    ;7513  LOOP.T2.1:
                    ;7514  WRITE1.DAR.T2:
U 0799, B6,4015     ;7515          MOV LS[#1] TO WR[0]                      ;GET DATA TO WRITE TO DAR
U 079A, 97,A415     ;7516          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]     ;WRITE A ONE TO IDC DISK
                    ;7517                                                   ; ADDRESS REGISTER
                    ;7518
U 079B, 36,9C15     ;7519          MOV LS[ZERO] TO WR[0]                    ;CHANGE THE DATA IN WR[0]
U 079C, 3F,3295     ;7520          MOV WR[1] TO LS[PORT0]                   ; AND IN LS[PORT0]
                    ;7521
U 079D, 90,9015     ;7522          MISC [SET.PORT.I.0]                      ;SELECT THE PORT TO BUS
U 079E, 17,8415     ;7523          PORT.READ PORT.ADRS[DAR]                 ;READ BACK THE IDC DAR
U 079F, 3B,3215     ;7524          MOV PORT TO LS[PORT0]                    ;MOV THE RECEIVED DATA TO LS
U 07A0, 37,3215     ;7525          MOV LS[PORT0] TO WR[0]                   ;MOV RECEIVED DATA FOR COMPARE
U 07A1, 36,4095     ;7526          MOV LS[#1] TO WR[1]                      ;GET EXPECTED DATA
                    ;7527          CMP WR[0] WITH WR[1],                    ;CMP THE EXP AND REC DATA
U 07A2, 26,40B5     ;7528           DT(LONG)&SET.ALU.CC
U 07A3, 08,7AF1     ;7529          JMP.IF[NEQ] TO [NO.IDC.T2]               ;GO PRINT NO IDC
                    ;7530
                    ;7531  WRITE0.DAR.T2:
U 07A4, 36,9C15     ;7532          MOV LS[#0] TO WR[0]                      ;GET DATA TO WRITE TO DAR
U 07A5, 97,A415     ;7533          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]     ;WRITE A ZERO TO IDC DISK
                    ;7534                                                   ; ADDRESS REGISTER
                    ;7535
U 07A6, B6,4015     ;7536          MOV LS[#1] TO WR[0]                      ;CHANGE THE DATA IN WR[0]
U 07A7, 3F,3295     ;7537          MOV WR[1] TO LS[PORT0]                   ; AND LS[PORT0]
```

```
                  ;7538
U 07A8, 90,9015   ;7539           MISC [SET.PORT.I.O]                     ;SELECT THE PORT TO BUS
U 07A9, 17,8415   ;7540           PORT.READ PORT.ADRS[DAR]                ;READ BACK THE IDC DAR
U 07AA, 3B,3215   ;7541           MOV PORT TO LS[PORTO]                   ;MOV THE RECEIVED DATA TO LS
U 07AB, 37,3215   ;7542           MOV LS[PORTO] TO WR[0]                  ;MOV RECEIVED DATA FOR COMPARE
U 07AC, B6,9C95   ;7543           MOV LS[#0] TO WR[1]                     ;GET EXPECTED DATA
                  ;7544           CMP WR[0] WITH WR[1],                   ;CMP THE EXP AND REC DATA
U 07AD, 26,40B5   ;7545            DT(LONG)&SET.ALU.CC
U 07AE, 88,7BD9   ;7546           JMP.IF[EQL] TO [IDC.FOUND.T2]           ;GO PRINT IDC FOUND
                  ;7547
                  ;7548   NO.IDC.T2:
U 07AF, 36,7E15   ;7549           MOV LS[PRINT.IDC] TO WR[0]              ;SET UP CONTROL STATUS WORD TO
U 07B0, 3E,8015   ;7550           MOV WR[0] TO LS[CONTROL.STATUS]         ; PRINT NO IDC FOUND
U 07B1, 10,E015   ;7551           MISC [SET.CP.ATTN]                      ;SET ATTENTION TO THE CONSOLE
                  ;7552   WAIT.T2.1:
U 07B2, 08,7B24   ;7553           JMP [WAIT.T2.1]                         ;WAIT FOR CONSOLE TO RETURN
                  ;7554                                                   ; CONTROL
                  ;7555
U 07B3, 36,9015   ;7556           MOV LS[ERROR.CONTROL] TO WR[0]          ;RUNNING UNDER APT?
                  ;7557           BIT LS[APT.PRESENT] WITH WR[0],
U 07B4, 59,4A35   ;7558            DT(LONG)&SET.ALU.CC
U 07B5, 88,E069   ;7559           JMP.IF[BITS.CLR] TO [END.SECTION]       ;IF APT NOT PRESENT THEN DO
                  ;7560                                                   ; NOT RUN DIAGNOSTIC
U 07B6, B6,6E15   ;7561           MOV LS[NA.EXP.REC] TO WR[0]
U 07B7, 3E,8015   ;7562           MOV WR[0] TO LS[CONTROL.STATUS]         ;SET UP TO FORCE AN ERROR
U 07B8, 2F,8015   ;7563           CLR WR[0]
U 07B9, 36,4095   ;7564           MOV LS[#1] TO WR[1]
U 07BA, 08,693C   ;7565           JSR [CHECK.RESULT]                      ;GO FORCE THE ERROR
U 07BB, 08,7994   ;7566           JMP [LOOP.T2.1]                         ;LOOP HERE IF ERR & LOE
                  ;7567
U 07BC, 08,E064   ;7568           JMP [END.SECTION]                       ;NO IDC AVAILABLE, GO TO END
                  ;7569                                                   ; OF SECTION
                  ;7570
                  ;7571   IDC.FOUND.T2:
U 07BD, 7F,0E15   ;7572           INC LS[T7.SUB]                          ;SET TO PRINT IDC FOUND
U 07BE, 36,7E15   ;7573           MOV LS[PRINT.IDC] TO WR[0]              ;SET CONTROL STATUS WORD TO
U 07BF, 3E,8015   ;7574           MOV WR[0] TO LS[CONTROL.STATUS]         ; PRINT MESSAGE
U 07C0, 10,E015   ;7575           MISC [SET.CP.ATTN]                      ;SET ATTENTION TO THE CONSOLE
                  ;7576   WAIT.T2.2:
U 07C1, 88,7C14   ;7577           JMP [WAIT.T2.2]                         ;WAIT FOR CONSOLE TO RETURN
                  ;7578
                  ;7579   END.T2:                                         ;END TEST
                  ;7580
```

```
.PAGE  "Test 3  DISK DRIVE CONFIGURATION SIZING TEST"
;++                         (not run in CRD mode)
;
; Test Description:
;
;         This test will check the configuration of disk drives attached to
;         the system. It will select drive 0 and check the status for R80/RL02
;         presence by looking at the DRDY bit of the CSR being read back from
;         the PORT. It will then select drive 1 and check for RL02. Anything
;         other than drive 0 = R80 or RL02 and drive 1 = RL02 will be treated
;         as an error.
;
; Assumption:
;
;         Both CPU micros and IDC part 1 have been run successfully.
;
; Test steps:
;
;           1. Move zero to WR[0]
;           2. Write the CSR with WR[0]
;           3. Read the CSR in WR[0]
;           4. Mask out all bits except <26 & 0>
;           5. Check (a) if bit<0>=1 for DRDY, if so
;           6.      Then check if bit<26>=1 for R80 present, if so
;           7.           Then set bit 8 in LS[DR0.R80]
;           8.           Else put 0 in LS[DR0.R80] and set bit 12 in LS[DR0.RL02]
;           9.      Else
;          10.           set up all related error parameters;
;          11.           and report ERROR 1 for no drive on 0
;          12. End check (a)
;          13. Load WR[0] with LS[BIT8] for drive 1 select bit
;          14. Write the CSR with WR[0]
;          15. Read the CSR into WR[0]
;          16. Mask out all bits except <26 & 0>
;          17. Check (b) if bit<0>=1 for DRDY, if so
;          18.      Then check if bit<26>=1 for R80 present, if so
;          19.           Then set up all related error parameters;
;          20.                 and report ERROR 2 for R80 present on drive 1
;          21.           Else set bit 13 in LS[DR1.RL02]
;          22.      Else
;          23.           set up related error parameters;
;          24.           and report ERROR 3 for no drive present on 1
;          25. End check (b)
;          26. Check if it is running APT from LS[APT.PRESENT], if so;
;          27.      Then see if hardware byte same as drive size result, if not;
;          28.                Then report ERROR 4
;          29.                Else,
;          30.      Else,
;          31. Test ends
;
; Errors:
;
;  ERROR 1 - Drive 0 does not have any disk drive
;
;  ERROR 2 - Drive 1 does not have a disk drive
```

```
                  ;
                  ;  ERROR 3 - Drive 1 does not have a RL02 disk drive
                  ;
                  ;              Printout:
                  ;                      EXPECTED        RECEIVED        MODULE
                  ;                      N/A             N/A             IDC
                  ;
                  ;  *** ERROR 4 only occurs when diagnostic is run under APT ***
                  ;
                  ;  ERROR 4 - Drive sizing result does not agree with drive configuration
                  ;            under APT
                  ;
                  ; Debug:
                  ;
                  ;       If either one of the errors occur even if a correct drive up and
                  ;       spinning is known to be attached to the system, then the DRIVE
                  ;       SEL, RL DRIVE SELECT & DRIVE READY signals should be checked for
                  ;       faults.
                  ;
                  ;--

                  T.3:
U 07C2, B6,5E15           MOV LS[BEGIN.TEST] TO WR[0]              ;set bit 15 in WR[0] for
                                                                   ; control and status word
U 07C3, 3E,8015           MOV WR[0] TO LS[CONTROL.STATUS]          ;set bit 15 in control and
                                                                   ; status word - bit 15
                                                                   ; indicates start of test to
                                                                   ; console
U 07C4, 10,E015           MISC [SET.CP.ATTN]                       ;issue CPU ATTN to console

                  WAIT.T3.0:
U 07C5, 08,7C54           JMP [WAIT.T3.0]                          ;loop here waiting for
                                                                   ; console to respond
U 07C6, 65,A015           CLR LS[PREVIOUS.ERROR]                   ;clear previous error number
                                                                   ; in LS
U 07C7, 17,CC15           PORT.CONTROL [CLEAR.IDC]                 ;init. IDC
U 07C8, 17,CC15           PORT.CONTROL [CLEAR.IDC]
U 07C9, 17,CC15           PORT.CONTROL [CLEAR.IDC]

                  T3.1:
U 07CA, 36,9C15           MOV LS[SETCSR.DS0] TO WR[0]              ;drive 0 select bit
U 07CB, 17,A015           PORT.WRITE PORT.ADRS[CSR] WITH WR[0]     ;write CSR to select drive 0

                  WAIT.T3.1:
U 07CC, 88,700C           JSR [WAIT.CSR.CRDY]                      ;wait for CSR to get ready
U 07CD, 5B,001E           SKIP.IF[SKIP]                            ;ready now
U 07CE, 08,7CC4           JMP [WAIT.T3.1]                          ;no, wait some more

U 07CF, 36,4095           MOV LS[#1] TO WR[1]                      ;set WR1 = 1
U 07D0, 3E,8295           MOV WR[1] TO LS[ERROR.NUMBER]            ;set up error number = 1 in LS
U 07D1, A0,C295           COM WR[1]                                ;create a constant #FFFFFFFE as
U 07D2, BE,8A95           MOV WR[1] TO LS[ERROR.MASK]              ; error mask for CSR<0>
U 07D3, 36,4A95           MOV LS[IDC] TO WR[1]                     ;store IDC module code in LS
U 07D4, BE,8C95           MOV WR[1] TO LS[MODULE.NUM]              ; to indicate module at fault
```

```
U 07D5, 90,9015  ;7691          MISC [SET.PORT.I.0]                      ;select the port to bus
U 07D6, 97,8015  ;7692          PORT.READ PORT.ADRS[CSR]                 ;read CSR back
U 07D7, 3B,3215  ;7693          MOV PORT TO LS[PORT0]                    ; from the port and
U 07D8, 37,3215  ;7694          MOV LS[PORT0] TO WR[0]                   ;  load it in WR0
U 07D9, 36,4095  ;7695          MOV LS[#1] TO WR[1]                      ;expected DRDY bit set
U 07DA, 08,693C  ;7696          JSR [CHECK.RESULT]                       ;is CSR DRDY set ??
U 07DB, 08,7CA4  ;7697          JMP [T3.1]                               ;loop on error if LOE is set
                 ;7698
                 ;7699  ;Yes, DRDY is set. i.e. there is a drive in drive 0.
                 ;7700
U 07DC, 37,3215  ;7701          MOV LS[PORT0] TO WR[0]                   ;reload CSR word in WR0
                 ;7702          BIT WR[0] WITH LS[BIT26],                ;tst R80 bit for R80 presence
U 07DD, D9,7435  ;7703           DT(LONG)&SET.ALU.CC
U 07DE, 08,7E41  ;7704          JMP.IF[BIT.SET] TO [SET.DR0.R80]         ;yes, do the next test
                 ;7705
U 07DF, 2F,8015  ;7706          CLR WR[0]
U 07E0, 3F,0C15  ;7707          MOV WR[0] TO LS[DR0.R80]                 ;put zero in LS[DR0.R80]
U 07E1, B6,5815  ;7708          MOV LS[BIT12] TO WR[0]                   ;set RL02 bit in APT H/W
U 07E2, BF,0E15  ;7709          MOV WR[0] TO LS[DR0.RL02]                ; byte format
U 07E3, 08,7E84  ;7710          JMP [T3.INIT.2]                          ;go see what drive 1 has
                 ;7711
                 ;7712  SET.DR0.R80:
U 07E4, 36,5015  ;7713          MOV LS[BIT8] TO WR[0]
U 07E5, 3F,0C15  ;7714          MOV WR[0] TO LS[DR0.R80]                 ;set LS[DR0.R80]
U 07E6, 2F,8015  ;7715          CLR WR[0]
U 07E7, BF,0E15  ;7716          MOV WR[0] TO LS[DR0.RL02]                ;clr. LS[DR0.RL02]
                 ;7717
                 ;7718  T3.INIT.2:
U 07E8, 2F,8015  ;7719          CLR WR[0]
U 07E9, BF,1015  ;7720          MOV WR[0] TO LS[DR1.RL02]                ;init. the RL02 presence indicator
U 07EA, 3F,8E15  ;7721          MOV WR[0] TO LS[WAIT.CNTR]               ;init. wait loop counter
                 ;7722
                 ;7723  T3.2:
U 07EB, 36,5015  ;7724          MOV LS[SETCSR.DS1] TO WR[0]              ;drive 1 select bit
U 07EC, 17,A015  ;7725          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]     ;write CSR to select dr. 1
                 ;7726
                 ;7727  WAIT.T3.2:
U 07ED, 88,700C  ;7728          JSR [WAIT.CSR.CRDY]                      ;wait for CSR to get ready
U 07EE, 5B,001E  ;7729          SKIP.IF[SKIP]                            ;ready now
U 07EF, 08,7ED4  ;7730          JMP [WAIT.T3.2]                          ;no, wait some more
                 ;7731
U 07F0, B6,4295  ;7732          MOV LS[#2] TO WR[1]                      ;load WR1 = 2
U 07F1, 3E,8295  ;7733          MOV WR[1] TO LS[ERROR.NUMBER]            ;set up error number = 2 in LS
U 07F2, 36,4095  ;7734          MOV LS[#1] TO WR[1]                      ;load WR = 1
U 07F3, A0,C295  ;7735          COM WR[1]                                ;create a constant #FFFFFFFE as
U 07F4, BE,8A95  ;7736          MOV WR[1] TO LS[ERROR.MASK]              ; the error mask for CSR<0>
U 07F5, 36,4A95  ;7737          MOV LS[IDC] TO WR[1]                     ;store IDC module code in LS
U 07F6, BE,8C95  ;7738          MOV WR[1] TO LS[MODULE.NUM]              ; to indicate module at fault
                 ;7739
U 07F7, 90,9015  ;7740          MISC [SET.PORT.I.0]                      ;select the port to bus
U 07F8, 97,8015  ;7741          PORT.READ PORT.ADRS[CSR]                 ;read CSR back
U 07F9, 3B,3215  ;7742          MOV PORT TO LS[PORT0]                    ; from the port and
U 07FA, 37,3215  ;7743          MOV LS[PORT0] TO WR[0]                   ;  load it in WR0
U 07FB, 36,4095  ;7744          MOV LS[#1] TO WR[1]                      ;expected DRDY bit set
U 07FC, 08,693C  ;7745          JSR [CHECK.RESULT]                       ;is CSR DRDY set ??
```

```
U 07FD, 08,7CA4   ;7746          JMP [T3.1]                                 ;loop on error if LOE is set
                  ;7747
                  ;7748  ;Yes, DRDY is set. i.e. there is a drive in drive 1.
                  ;7749
U 07FE, B6,C095   ;7750          MOV LS[#3] TO WR[1]                        ;load WR1 = 3
U 07FF, 3E,8295   ;7751          MOV WR[1] TO LS[ERROR.NUMBER]              ;set up error number = 3 in LS
U 0800, B6,7495   ;7752          MOV LS[BIT26] TO WR[1]                     ;load WR = 04000000
U 0801, A0,C295   ;7753          COM WR[1]                                  ;create a constant #FBFFFFFF as
U 0802, BE,8A95   ;7754          MOV WR[1] TO LS[ERROR.MASK]                ; the error mask for CSR<26>
U 0803, 36,4A95   ;7755          MOV LS[IDC] TO WR[1]                       ;store IDC module code in LS
U 0804, BE,8C95   ;7756          MOV WR[1] TO LS[MODULE.NUM]                ; to indicate module at fault
U 0805, 37,3215   ;7757          MOV LS[PORT0] TO WR[0]                     ;reload CSR word in WR0
U 0806, 2F,8295   ;7758          CLR WR[1]                                  ;expected value for CSR<26>
U 0807, 08,693C   ;7759          JSR [CHECK.RESULT]                         ;see if R80 present
U 0808, 08,7E84   ;7760          JMP [T3.INIT.2]                            ;loop on error if LOE is set
                  ;7761
U 0809, 36,5A15   ;7762          MOV LS[BIT13] TO WR[0]                     ;set bit 13 in LS[DR1.RL02]
U 080A, BF,1015   ;7763          MOV WR[0] TO LS[DR1.RL02]                  ; for APT H/W byte format
                  ;7764
                  ;7765  ;Check if the diagnostic is run under APT :
                  ;7766
U 080B, 36,9015   ;7767          MOV LS[ERROR.CONTROL] TO WR[0]             ;look at the error control word
                  ;7768          BIT LS[APT.PRESENT] WITH WR[0],            ;see if it is running APT
U 080C, 59,4A35   ;7769           DT(LONG)&SET.ALU.CC
U 080D, 08,81C9   ;7770          JMP.IF[BITS.CLR] TO [END.T3]               ;no, test ends
                  ;7771
                  ;7772  ;Set up error parameters for ERROR 4 :
                  ;7773
U 080E, 36,4415   ;7774          MOV LS[#4] TO WR[0]                        ;load WR1 = 4
U 080F, BE,8215   ;7775          MOV WR[0] TO LS[ERROR.NUMBER]              ;set up error number = 4 in LS
U 0810, 37,C815   ;7776          MOV LS[#FFFF00FF] TO WR[0]                 ;mask for APT H/W byte
U 0811, 3E,8A15   ;7777          MOV WR[0] TO LS[ERROR.MASK]                ; as the error mask
U 0812, 2F,8015   ;7778          CLR WR[0]                                  ;clr. module code
U 0813, 3E,8C15   ;7779          MOV WR[0] TO LS[MODULE.NUM]                ; for module number
                  ;7780
                  ;7781  ;Running APT, this will check if the drive sizing result equals the expected
                  ;7782  ; drive configuration as specified by the hardware byte :
                  ;7783
U 0814, B7,0C15   ;7784          MOV LS[DR0.R80] TO WR[0]                   ;fetch all the
U 0815, B7,0E95   ;7785          MOV LS[DR0.RL02] TO WR[1]                  ; drive sizing
U 0816, AE,C215   ;7786          BIS WR[1] TO WR[0]                         ;  result
U 0817, B7,1095   ;7787          MOV LS[DR1.RL02] TO WR[1]                  ;   as received
U 0818, AE,C215   ;7788          BIS WR[1] TO WR[0]                         ;    data
U 0819, 36,9295   ;7789          MOV LS[APT.PARAM] TO WR[1]                 ;H/W byte as expected
U 081A, 08,693C   ;7790          JSR [CHECK.RESULT]                         ;are they equal ??
U 081B, 08,8C54   ;7791          JMP [T.6.1]                                ;err and if LOE
                  ;7792
                  ;7793  END.T3:                                            ;end of test
                  ;7794
```

```
.PAGE  "Test 4  RL GET STATUS FUNCTION TEST (not run in CRD mode)"
;++
;
; Test Description:
;
;       This test will check the RL GET STATUS function by first write to
;       the DAR word with the marker, reset and get status bits set and then
;       write to the CSR with the drive 1 select <bit 8> and the get status
;       function <bit 2>. Then the 2-byte status of drive 1 will be read
;       from FIFO A and compared with the expected status as follows :
;
;                   RL02
;                   status
;                   bits       functions          expected states
;                   ------     ---------          ---------------
;
;                   <2-0>      state              101 lock on
;                     <3>      brush home           1 brush off disk
;                     <4>      heads out            1 heads on disk
;                     <5>      cover open           0 cover closed
;                     <6>      head select          0 upper head selected
;                     <7>      not used             x don't care (masked out)
;                     <8>      drive sel. err.      0 no drive select error
;                     <9>      volume check         0 cartridge not changed
;                    <10>      write gate err.      0 no write gate error
;                    <11>      spin err.            0 spindle speed correct
;                    <12>      seek timeout         0 seek within time
;                    <13>      write lock           0 write enabled
;                    <14>      current in -         0 heads without current -
;                               head error             (i.e. not doing a R/W)
;                    <15>      write data err.      0 not doing a write
;
;
; Assumption:
;
;       Both CPU micros and IDC part 1 has been run successfully
;
; Test Steps:
;
;         1. Load WR[0] with bits <3, 1 & 0>
;         2. Write DAR with WR[0] (i.e. the marker, get status and reset bits)
;         3. Load WR[0] with bits <8 & 2>
;         4. Write CSR with WR[0] (i.e. drive 1 select and get status bits)
;         5. Set up a time out loop to wait for CRDY set
;         6. If time expires on waiting;
;         7.     Then report ERROR 1 for timeout on Get Status
;         8.     Else Read the drive status from FIFO A with WR[0]
;         9. Set up the error mask with LS[#FFFF0080]
;        10. Put LS[IDC] & LS[RL02] in LS[MODULE.NUM]
;        11. Load WR[1] with exp. data from LS[#0000001D]
;        12. Use JSR [check.result] to compare disk status
;        13. If error, report ERROR 2 for wrong drive status
;        14. Else test ends
;
; Error:
```

```
                   ;
                   ; ERROR 1 - Timeout error waiting for RL02 Get Status command to finish
                   ;
                   ;               Printout:
                   ;                       EXPECTED        RECEIVED        MODULE
                   ;                       N/A             N/A             IDC
                   ;
                   ; ERROR 2 - Disk drive 1 operational status incorrect
                   ;
                   ;               Printout:
                   ;                       EXPECTED        RECEIVED        MODULE
                   ;                       RL02 status     RL02 status     IDC/RL02
                   ;
                   ; Debug:
                   ;
                   ;   ERROR 1: Check RL WRITE CLOCK for fault which asserts on the CURRENT
                   ;            CLOCK & SEQUENCE CLOCK to sync RL02 drive with IDC; and also
                   ;            RL STATUS CLOCK which syncs the status of the RL02 disk drive
                   ;            asserted to the IDC.
                   ;
                   ;   ERROR 2: Check RL STATUS input to the RL02 receivers, DS DATA and its
                   ;            data gate, the disk data mux and data synchronizer. If all
                   ;            these check out O.K, the fault could possibly come from the
                   ;            drive itself.
                   ;
                   ;--

                   T.4:
U 081C, B6,5E15            MOV LS[BEGIN.TEST] TO WR[0]             ;set bit 15 in WR[0] for
                                                                   ; control and status word
U 081D, 3E,8015            MOV WR[0] TO LS[CONTROL.STATUS]         ;set bit 15 in control and
                                                                   ; status word - bit 15
                                                                   ; indicates start of test to
                                                                   ; console
U 081E, 10,E015            MISC [SET.CP.ATTN]                      ;issue CPU ATTN to console

                   WAIT.T4.0:
U 081F, 88,81F4            JMP [WAIT.T4.0]                         ;loop here waiting for
                                                                   ; console to respond
U 0820, 65,A015            CLR LS[PREVIOUS.ERROR]                  ;clear previous error number
                                                                   ; in LS
U 0821, 17,CC15            PORT.CONTROL [CLEAR.IDC]                ;init. IDC
U 0822, 17,CC15            PORT.CONTROL [CLEAR.IDC]
U 0823, 17,CC15            PORT.CONTROL [CLEAR.IDC]

                   WAIT.CRDY.T4.1:
U 0824, 88,700C            JSR [WAIT.CSR.CRDY]                     ;check if CRDY after IDC init.
U 0825, 08,8274            JMP [T.4.1]                             ;yes, CRDY is set
U 0826, 08,8244            JMP [WAIT.CRDY.T4.1]                    ;no, wait some more

                   T.4.1:
U 0827, 37,C415            MOV LS[0B] TO WR[0]                     ;DAR format for get status
U 0828, 97,A415            PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ;write DAR with the command

U 0829, B7,0295            MOV LS[GET.STATUS] TO WR[1]             ;CSR format for get status
```

```
U 082A, 36,5015  ;7905          MOV LS[SETCSR.DS1] TO WR[0]                ;select drive 1
U 082B, AE,C095  ;7906          BIS WR[0] TO WR[1]
U 082C, 97,A095  ;7907          PORT.WRITE PORT.ADRS[CSR] WITH WR[1]       ;write to the CSR
                 ;7908                                                     ;MAINT=0,CRDY=0,F<2:0>=010
                 ;7909
                 ;7910  ;*** RL02 GET STATUS IDC FUNCTION CODES : ***
                 ;7911
                 ;7912  ;-----------------------------------------------------------------
                 ;7913  ;=1
                 ;7914  ;
                 ;7915  ;TEST.OR.RUN:
                 ;7916  ;
                 ;7917  ;       BEN2/MAINT,             ;IS THAT YOU, VICKIE?
                 ;7918  ;       NAD/DRIVE.TYPE
                 ;7919  ;
                 ;7920  ;-----------------------------------------------------------------
                 ;7921  ;=0**
                 ;7922  ;
                 ;7923  ;DRIVE.TYPE:
                 ;7924  ;
                 ;7925  ;       BEN0/R80,               ;WHICH TYPE OF DRIVE WAS SELECTED?
                 ;7926  ;       NAD/RL02.IFORK
                 ;7927  ;
                 ;7928  ;-----------------------------------------------------------------
                 ;7929  ;=0
                 ;7930  ;
                 ;7931  ;RL02.IFORK:
                 ;7932  ;
                 ;7933  ;       UCMD/SET.FIFOA,
                 ;7934  ;       FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
                 ;7935  ;       BEN0/F0,                ;8 WAY INSTRUCTION FORK.
                 ;7936  ;       BEN1/F1,                ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                 ;7937  ;       BEN2/F2,
                 ;7938  ;       NAD/NOP.CMD
                 ;7939  ;
                 ;7940  ;-----------------------------------------------------------------
                 ;7941  ;=010
                 ;7942  ;
                 ;7943  ;GET.STAT.CMD:
                 ;7944  ;
                 ;7945  ;       NAD/SETUP.FIFO2         ;START OF THE RL02 GET STATUS COMMAND.
                 ;7946  ;
                 ;7947  ;-----------------------------------------------------------------
                 ;7948  ;
                 ;7949  ;SETUP.FIFO2:
                 ;7950  ;
                 ;7951  ;       UCMD/SET.FIFOA, ;SELECT FIFOA AND CLEAR ADDRESS CNTR.
                 ;7952  ;       CLR.FIFO.ADR
                 ;7953  ;
                 ;7954  ;-----------------------------------------------------------------
                 ;7955  ;
                 ;7956  ;CHGCLK2:
                 ;7957  ;
                 ;7958  ;       ENB.CLR.BC/ON,          ;CLEAR BC SO THAT DARO IS SELECTED
                 ;7959  ;       ENB.SYS.CLK             ;USE RL02 WRITE CLOCK
```

```
;7960   ;
;7961   ;-------------------------------------------------------------------
;7962   ;
;7963   ;SETUP.DRV.CMD2:
;7964   ;
;7965   ;           ENB.CLR.BC/ON,              ;PREPARE TO SEND COMMAND.
;7966   ;           LOAD.LOOP.CNTR.[0F0]
;7967   ;
;7968   ;-------------------------------------------------------------------
;7969   ;=0
;7970   ;
;7971   ;SEND.DRV.CMD2:
;7972   ;
;7973   ;           ENB.DRV.CMD/ON,             ;SHIFT OUT GET STATUS COMMAND
;7974   ;           INCR.LPCNTR/ON,
;7975   ;           BENO/CNTR.OVFLW,
;7976   ;           NAD/SEND.DRV.CMD2
;7977   ;
;7978   ;-------------------------------------------------------------------
;7979   ;
;7980   ;CHGCLK2A:
;7981   ;
;7982   ;           ENB.STATUS.CLK,             ;RUN OFF THE STATUS CLOCK
;7983   ;           LOAD.LOOP.CNTR.[0FA]        ; Fix for new version
;7984   ;
;7985   ;-------------------------------------------------------------------
;7986   ;=0
;7987   ;
;7988   ;READ.STATUS2:
;7989   ;
;7990   ;           INCR.LPCNTR/ON,             ;WHEN SEQ CLK STARTS, THERE WILL ALREADY
;7991   ;           BENO/CNTR.OVFLW,            ;BE 3 STATUS BITS IN THE DSR.
;7992   ;           NAD/READ.STATUS2
;7993   ;
;7994   ;-------------------------------------------------------------------
;7995   ;=1
;7996   ;
;7997   ;STORE.HIBYTE2:
;7998   ;
;7999   ;           ENB.DSR/ON,                 ;8TH STATUS BIT
;8000   ;           WRT.FIFO/ON,                ;LOAD BYTE INTO FIFO
;8001   ;           LOAD.LOOP.CNTR.[0F9],
;8002   ;           INCR.FIFO.CNTR/ON
;8003   ;
;8004   ;-------------------------------------------------------------------
;8005   ;=0
;8006   ;
;8007   ;READ.STATUS2A:
;8008   ;
;8009   ;           INCR.LPCNTR/ON,             ;SHIFT IN 2ND BYTE
;3010   ;           BENO/CNTR.OVFLW,
;8011   ;           NAD/READ.STATUS2A
;8012   ;
;8013   ;-------------------------------------------------------------------
;8014   ;=1
```

```
;8015   ;
;8016   ;STORE.LOBYTE2:
;8017   ;
;8018   ;       ENB.DSR/ON,             ;LAST STATUS BIT
;8019   ;       WRT.FIFO/ON             ;LOAD BYTE INTO FIFO
;8020   ;
;8021   ;-----------------------------------------------------------------------
;8022   ;
;8023   ;END.STAT.CMD2:
;8024   ;
;8025   ;       ENB.CPU.CLK,            ;BACK TO THE NEBULA CLOCK.
;8026   ;       CLR.FIFO.ADR,           ;CLEAR THE FIFO ADDRESS COUNTER
;8027   ;       NAD/XFER.DONE4
;8028   ;
;8029   ;-----------------------------------------------------------------------
;8030   ;
;8031   ;XFER.DONE4:
;8032   ;
;8033   ;       UCMD/SET.CRDY,          ;SIGNAL END OF TRANSFER.
;8034   ;       FUNC/SET.INT,
;8035   ;       NAD/SET.INT.AND.IDLE
;8036   ;
;8037   ;-----------------------------------------------------------------------
;8038   ;=111111110
;8039   ;
;8040   ;SET.INT.AND.IDLE:
;8041   ;
;8042   ;       FUNC/SET.INT,           ; The purpose of this state is to give CRDY one
;8043   ;       NAD/IDLE                ; state to reset before asserting interrupt.
;8044   ;                               ; CRDY is now used in the INT equation.
;8045   ;
;8046   ;-----------------------------------------------------------------------
;8047   ;=1
;8048   ;
;8049   ;RTN.TO.IDLE:
;8050   ;IDLE:
;8051   ;
;8052   ;       BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
;8053   ;       NAD/SETUP.DRV.CHK
;8054   
;8055   ;-----------------------------------------------------------------------
;8056   ;=0
;8057   ;
;8058   ;SETUP.DRV.CHK:
;8059   ;
;8060   ;       LOAD.LOOP.CNTR.[OFB],   ;COUNTER WILL BE USED
;8061   ;       NAD/SEL.NXT.DRV         ;TO SEQUENCE THRU DRIVES.
;8062   ;
;8063   ;-----------------------------------------------------------------------
;8064   ;
;8065   ;SEL.NXT.DRV:
;8066   ;
;8067   ;       SEL.DRV/UPROG,
;8068   ;       BENO/CNTR.OVFLW,
;8069   ;       NAD/CHK.DRV.STATUS
```

```
                ;8070  ;
                ;8071  ;------------------------------------------------------------------
                ;8072  ;=0
                ;8073  ;
                ;8074  ;CHK.DRV.STATUS:
                ;8075  ;        ; Give drive status time to settle down.
                ;8076  ;
                ;8077  ;        SEL.DRV/UPROG,          ;SELECT DRIVE
                ;8078  ;        NAD/DRV.STATUS.DISPATCH
                ;8079  ;
                ;8080  ;------------------------------------------------------------------
                ;8081  ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
                ;8082  ;
                ;8083  ;DRV.STATUS.DISPATCH:
                ;8084  ;        SEL.DRV/UPROG,          ;SELECT DRIVE
                ;8085  ;        BEN1/DRIVE.ONLINE,      ;AND BRANCH ON STATUS.
                ;8086  ;        BEN0/DRIVE.RDY,
                ;8087  ;        NAD/-ERR.-OL.-RDY
                ;8088  ;
                ;8089  ;
                ;8090  ;------------------------------------------------------------------
                ;8091  ;=011
                ;8092  ;
                ;8093  ;-ERR.OL.RDY:
                ;8094  ;
                ;8095  ;        SEL.DRV/UPROG,          ;NOP
                ;8096  ;        INCR.LPCNTR/ON,         ;ADD 1 TO DRIVE SELECT.
                ;8097  ;        NAD/SEL.NXT.DRV
                ;8098  ;
                ;8099  ;------------------------------------------------------------------
                ;8100  ;
                ;8101  ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                ;8102  ;
                ;8103  ;               UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                ;8104  ;
                ;8105  ;------------------------------------------------------------------
                ;8106  ;
                ;8107  ;=1
                ;8108  ;
                ;8109  ;RTN.TO.IDLE:
                ;8110  ;IDLE:
                ;8111  ;
                ;8112  ;        BEN0/CRDY.L,            ;LOOK FOR GO (CRDY L)
                ;8113  ;        NAD/SETUP.DRV.CHK
                ;8114  ;
                ;8115  ;------------------------------------------------------------------
                ;8116
                ;8117  ;Set up parameters for time-out error report :
                ;8118
U 082D, 36,4095 ;8119          MOV LS[01] TO WR[1]
U 082E, 3E,8295 ;8120          MOV WR[1] TO LS[ERROR.NUMBER]          ;set up error number = 1 in LS
U 082F, E5,8A15 ;8121          CLR LS[ERROR.MASK]                     ;no error mask used here
U 0830, 36,4A95 ;8122          MOV LS[IDC] TO WR[1]                   ;store module code in LS to
U 0831, BE,8C95 ;8123          MOV WR[1] TO LS[MODULE.NUM]            ; modules
U 0832, 36,6E95 ;8124          MOV LS[NA.EXP.REC] TO WR[1]            ;put "N/A" for printout data
```

```
U 0833, BE,8095   ;8125           MOV WR[1] TO LS[CONTROL.STATUS]
                  ;8126
U 0834, 2F,8015   ;8127           CLR WR[0]                               ;Initialize time-out counter
U 0835, BF,8C15   ;8128           MOV WR[0] TO LS[TIMEOUT.CNTR]
                  ;8129
                  ;8130   ;Timeout loop to wait for CRDY after get status has been issued.
                  ;8131
                  ;8132   WAIT.IDC.T4.1:
U 0836, 37,8C15   ;8133           MOV LS[TIMEOUT.CNTR] TO WR[0]           ;load the current time count
U 0837, 37,9095   ;8134           MOV LS[WAIT.TIME1] TO WR[1]             ;time limit on get status
                  ;8135           CMP WR[0] WITH WR[1],                   ;timeout yet ???
U 0838, 26,40B5   ;8136            DT(LONG)&SET.ALU.CC
U 0839, 88,83F9   ;8137           JMP.IF[EQL] TO [T4.ERR.1]               ;yes, timeout on get status
U 083A, 20,4015   ;8138           INC WR[0]                               ;inc. time count
U 083B, BF,8C15   ;8139           MOV WR[0] TO LS[TIMEOUT.CNTR]
U 083C, 88,700C   ;8140           JSR [WAIT.CSR.CRDY]                     ;check if CRDY is set
U 083D, 08,8444   ;8141           JMP [IDC.DONE.T4.1]                     ;yes, CRDY is set now
U 083E, 08,8364   ;8142           JMP [WAIT.IDC.T4.1]                     ;no, wait some more
                  ;8143
                  ;8144   ;Force an error in the CHECK.RESULT routine for this timout :
                  ;8145
                  ;8146   T4.ERR.1:
U 083F, 2F,8015   ;8147           CLR WR[0]                               ;set unequal values
U 0840, 36,4095   ;8148           MOV LS[#1] TO WR[1]                     ; between WR0 and WR1
U 0841, 08,693C   ;8149           JSR [CHECK.RESULT]                      ;force an error
U 0842, 08,8274   ;8150           JMP [T.4.1]                             ;if LOE is set
U 0843, 08,8444   ;8151           JMP [IDC.DONE.T4.1]
                  ;8152
                  ;8153   ;Set up error parameters for ERROR 2 report :
                  ;8154
                  ;8155   IDC.DONE.T4.1:
U 0844, 36,4215   ;8156           MOV LS[#2] TO WR[0]                     ;error number = 2
U 0845, BE,8215   ;8157           MOV WR[0] TO LS[ERROR.NUMBER]
U 0846, 37,A415   ;8158           MOV LS[#FFFF0080] TO WR[0]              ;error mask for RL02
U 0847, 3E,8A15   ;8159           MOV WR[0] TO LS[ERROR.MASK]             ; expected status
U 0848, 36,4A95   ;8160           MOV LS[IDC] TO WR[1]                    ;store IDC and RL02 in LS
U 0849, 47,4C95   ;8161           BIS LS[RL02] TO WR[1]                   ; as module codes to indicate
U 084A, BE,8C95   ;8162           MOV WR[1] TO LS[MODULE.NUM]             ;  modules at fault
U 084B, E5,8015   ;8163           CLR LS[CONTROL.STATUS]                  ;clear [NA.EXP.REC] bit
                  ;8164
                  ;8165   ;Read the current drive status from FIFO A :
                  ;8166
U 084C, 17,D815   ;8167           PORT.CONTROL [SEL.FIFO.A]               ;select FIFO A
U 084D, 17,C015   ;8168           PORT.CONTROL [CLEAR.FIFO.CNTR]          ;clear the FIFO A adr cntr.
U 084E, 90,9015   ;8169           MISC [SET.PORT.I.O]                     ;select IDC
U 084F, 97,8C15   ;8170           PORT.READ PORT.ADRS[DATA.WORD]          ;read status from FIFO A
U 0850, 38,3215   ;8171           MOV PORT TO LS[PORT0]
U 0851, 37,3215   ;8172           MOV LS[PORT0] TO WR[0]                  ;received drive status
U 0852, 37,C695   ;8173           MOV LS[#1D] TO WR[1]                    ;expected status
U 0853, 08,693C   ;8174           JSR [CHECK.RESULT]                      ;is this the right status ??
U 0854, 08,8444   ;8175           JMP [IDC.DONE.T4.1]                     ;if LOE is set
U 0855, 08,8564   ;8176           JMP [END.T4]                            ;test done !!
                  ;8177
                  ;8178   END.T4:                                         ;end of test
```

```
;8179  .PAGE  "Test 5  RL READ HEADER AND SEEK TESTS (run in all modes - except CRD mode)"
;8180  ;++
;8181  ;
;8182  ; Test Description:
;8183  ;
;8184  ;       This test will check the RL READ HEADER function by first finding
;8185  ;       out where the heads are located : (write to the CSR with drive 1
;8186  ;       select <bit 8> and the read header function <bit 3>. Then the header
;8187  ;       information will be read from FIFO A). Next, a seek function to a
;8188  ;       predetermined cylinder will be performed : (the cylinder difference
;8189  ;       which will be loaded to the DAR with the marker, direction and
;8190  ;       cylinder difference bits; then CSR will be written with drive select
;8191  ;       1 <bit 8> and the seek function <bits 1 and 2>). Then to verify that
;8192  ;       both read header and seek functions work, a read header command will
;8193  ;       be once again issued to the drive to obtain the current cylinder
;8194  ;       address which will be compared against the expected address. This
;8195  ;       test will seek to 2 different cylinders on the disk: one for the
;8196  ;       innermost cylinder, another for the outermost.
;8197  ;
;8198  ; Assumption:
;8199  ;
;8200  ;       Both CPU micros and IDC part I has been run successfully and disk
;8201  ;       drive 1 is up and spinning and ready to accept commands.
;8202  ;
;8203  ; Test Steps:
;8204  ;
;8205  ;         1. Write CSR with drive 1 select and read header function bits
;8206  ;         2. Setup a timeout loop to wait for CRDY to set
;8207  ;         3. If timeout on read header command;
;8208  ;         4.      Then setup error parameters and report ERROR 1
;8209  ;         5.      Else port select FIFO A and clear its address counter
;8210  ;         6. Read header from FIFO A
;8211  ;         7. Mask the header with LS[#FFFFFF80] to get the cylinder address
;8212  ;         8. Write DAR with marker bit set and dir. bit set and the cylinder
;8213  ;            address difference (max. cyl. minus current cyl. adr.)
;8214  ;         9. Write CSR with drive 1 select and seek function bits
;8215  ;        10. Setup a timeout loop to wait for CRDY to set
;8216  ;        11. If timeout on seek command;
;8217  ;        12.   Then setup error parameters and report ERROR 2
;8218  ;        13.   Else write CSR with drive 1 select and read header function bits
;8219  ;        14. Wait for CRDY to set
;8220  ;        15. Port select FIFO A and clear its address counter
;8221  ;        16. Read header from FIFO A to WR[1]
;8222  ;        17. Get the cylinder difference by ANDing LS[#FFFFFF80] to WR[1]
;8223  ;        18. Compare the cylinder address with the max. cylinder address
;8224  ;        19. If non-compare setup error parameters and report ERROR 3
;8225  ;        20. Else write DAR with marker and dir. bits reset and the cylinder
;8226  ;             difference (difference between current cyl. and cyl. 0)
;8227  ;        21. Write CSR with drive 1 select and seek function bits
;8228  ;        22. Wait for CRDY to set
;8229  ;        23. Write CSR with drive 1 select and read header function bits
;8230  ;        24. Wait for CRDY to set
;8231  ;        25. Port select FIFO A and clear its address counter
;8232  ;        26. Read header from FIFO A to WR[1]
;8233  ;        27. Get the cylinder difference by ANDing LS[#FFFFFF80] to WR[1]
```

```
                  ;         28. Compare the cylinder address with zero
                  ;         29. If non-compare setup error parameters and report ERROR 4
                  ;         30. Else test ends
                  ;
                  ; Errors:
                  ;
                  ;  ERROR 1 - Timeout on a read header command
                  ;
                  ;  ERROR 2 - Timeout on a seek command
                  ;
                  ;         Printout:
                  ;              EXPECTED      RECEIVED        MODULE
                  ;              N/A           N/A             IDC
                  ;
                  ;  ERROR 3 - The R/W heads did not seek to the innermost cylinder
                  ;
                  ;  ERROR 4 - The R/W heads did not seek to the outermost cylinder
                  ;
                  ;         Printout:
                  ;              EXPECTED      RECEIVED        MODULE
                  ;              Cyl. adr.     cyl. adr.       RL02
                  ;
                  ; Debug:
                  ;
                  ;  ERROR 1: Check RL DRIVE READY, RL SYSTEM CLOCK, RL SECTOR PLS,
                  ;            RL SEC PLS & RL READ DATA signals which might have
                  ;            triggered the timeout if at fault.
                  ;
                  ;  ERROR 2: Check RL DRIVE READY, RL SYSTEM CLOCK & RL SECTOR PLS
                  ;            for fault.
                  ;
                  ;  ERROR 3 & 4: Check DAR serializer, RL DRIVE COMMAND and the driver
                  ;                where this signal comes from.
                  ;
                  ;--
                  T.5:
U 0856, B6,5E15           MOV LS[BEGIN.TEST] TO WR[0]             ;set bit 15 in WR[0] for
                                                                  ; control and status word
U 0857, 3E,8015           MOV WR[0] TO LS[CONTROL.STATUS]         ;set bit 15 in control and
                                                                  ; status word - bit 15
                                                                  ; indicates start of test to
                                                                  ; console
U 0858, 10,E015           MISC [SET.CP.ATTN]                      ;issue CPU ATTN to console

                  WAIT.T5.0:
U 0859, 08,8594           JMP [WAIT.T5.0]                         ;loop here waiting for
                                                                  ; console to respond

U 085A, 65,A015           CLR LS[PREVIOUS.ERROR]                  ;clear previous error number
                                                                  ; in LS
U 085B, 17,CC15           PORT.CONTROL [CLEAR.IDC]                ;init. IDC
U 085C, 17,CC15           PORT.CONTROL [CLEAR.IDC]
U 085D, 17,CC15           PORT.CONTROL [CLEAR.IDC]

                  ;Locate the heads on the disk by issuing a read header command.
```

```
                   ;8289
                   ;8290   T.5.2:
U 085E, B7,0615    ;8291           MOV LS[READ.HEADER] TO WR[0]              ;get the read_header command
U 085F, 47,5015    ;8292           BIS LS[SETCSR.DS1] TO WR[0]               ;select drive 1
U 0860, 17,A015    ;8293           PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;write the CSR
                   ;8294                                                     ;MAINT=0,CRDY=0,F<2:0>=100
                   ;8295
                   ;8296   ;*** RL02 READ HEADER IDC FUNCTION CODES : ***
                   ;8297
                   ;8298   ;-----------------------------------------------------------------------------
                   ;8299   ;=1
                   ;8300   ;
                   ;8301   ;TEST.OR.RUN:
                   ;8302   ;
                   ;8303   ;       BEN2/MAINT,               ;IS THAT YOU, VICKIE?
                   ;8304   ;       NAD/DRIVE.TYPE
                   ;8305   ;
                   ;8306   ;-----------------------------------------------------------------------------
                   ;8307   ;=0**
                   ;8308   ;
                   ;8309   ;DRIVE.TYPE:
                   ;8310   ;
                   ;8311   ;       BEN0/R80,                 ;WHICH TYPE OF DRIVE WAS SELECTED?
                   ;8312   ;       NAD/RL02.IFORK
                   ;8313   ;
                   ;8314   ;-----------------------------------------------------------------------------
                   ;8315   ;=0
                   ;8316   ;
                   ;8317   ;RL02.IFORK:
                   ;8318   ;
                   ;8319   ;       UCMD/SET.FIFOA,
                   ;8320   ;       FUNC/CLR.ECC.ERROR,       ; Enable clear of ECC/CRC errors for new function
                   ;8321   ;       BEN0/F0,                  ;8 WAY INSTRUCTION FORK.
                   ;8322   ;       BEN1/F1,                  ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                   ;8323   ;       BEN2/F2,
                   ;8324   ;       NAD/NOP.CMD
                   ;8325   ;
                   ;8326   ;-----------------------------------------------------------------------------
                   ;8327   ;=100
                   ;8328   ;
                   ;8329   ;RD.HDR.CMD:
                   ;8330   ;
                   ;8331   ;       NAD/WAIT.DRDY4            ;START OF RL02 READ HEADER COMMAND.
                   ;8332   ;
                   ;8333   ;-----------------------------------------------------------------------------
                   ;8334   ;=0
                   ;8335   ;
                   ;8336   ;WAIT.DRDY4:
                   ;8337   ;
                   ;8338   ;       BEN0/DRIVE.RDY,           ;WAIT FOR DRIVE READY
                   ;8339   ;       NAD/WAIT.DRDY4
                   ;8340   ;
                   ;8341   ;-----------------------------------------------------------------------------
                   ;8342   ;=1
                   ;8343   ;
```

```
;8344  ;CHGCLK4:
;8345  ;
;8346  ;       ENB.SYS.CLK             ;USE RL02 WRITE CLOCK
;8347  ;
;8348  ;--------------------------------------------------------------------
;8349  ;=0*
;8350  ;
;8351  ;WAIT.SEC.PLS4:
;8352  ;
;8353  ;       BEN1/SECTOR.PLS,        ;WAIT FOR LEADING EDGE OF SECTOR PULSE
;8354  ;       NAD/WAIT.SEC.PLS4
;8355  ;
;8356  ;--------------------------------------------------------------------
;8357  ;=1*
;8358  ;
;8359  ;SEC.PLS.FND4:
;8360  ;
;8361  ;       CLR.FIFO.ADR,
;8362  ;       NAD/WAIT.END.SEC.PLS4   ;SECTOR PULSE HAS BEEN FOUND
;8363  ;
;8364  ;--------------------------------------------------------------------
;8365  ;=1*
;8366  ;
;8367  ;WAIT.END.SEC.PLS4:
;8368  ;
;8369  ;       BEN1/SECTOR.PLS,        ;WAIT FOR END OF SECTOR PULSE
;8370  ;       NAD/WAIT.END.SEC.PLS4
;8371  ;
;8372  ;--------------------------------------------------------------------
;8373  ;=1
;8374  ;
;8375  ;SETUP.RD.HDR4:
;8376  ;
;8377  ;       LOOP.LOCK/ON,           ;SETUP INITIAL CONDITIONS
;8378  ;       FUNC/CLR.DSR,           ;FOR READING HEADER.
;8379  ;       LOAD.LOOP.CNTR.[0F9]
;8380  ;
;8381  ;--------------------------------------------------------------------
;8382  ;
;8383  ;CHGCLK4A:
;8384  ;
;8385  ;       ENB.DSK.CLK,            ;THE MICROSEQUENCER WILL STALL
;8386  ;       LOOP.LOCK/ON,           ;UNTIL SYNC SEEN IS SET (SYNC BIT FOUND)
;8387  ;       GEN.SYNC.[80]
;8388  ;
;8389  ;--------------------------------------------------------------------
;8390  ;=0
;8391  ;
;8392  ;READ.HDR.BYTE4:
;8393  ;
;8394  ;       LOOP.LOCK/ON,           ;READ FIRST HEADER BYTE
;8395  ;       INCR.LPCNTR/ON,
;8396  ;       BEN0/CNTR.OVFLW,
;8397  ;       NAD/READ.HDR.BYTE4
;8398  ;
```

```
;8399   ;-----------------------------------------------------------------------------------
;8400   ;=1
;8401   ;
;8402   ;STORE.HDR.BYTE4:
;8403   ;
;8404   ;       LOOP.LOCK/ON,           ;STORE FIRST HDR BYTE IN FIFO
;8405   ;       ENB.DSR/ON,
;8406   ;       WRT.FIFO/ON,
;8407   ;       INCR.FIFO.CNTR/ON,
;8408   ;       LOAD.LOOP.CNTR.[0F9]
;8409   ;
;8410   ;-----------------------------------------------------------------------------------
;8411   ;=0
;8412   ;
;8413   ;READ.HDR.BYTE4A:
;8414   ;
;8415   ;       LOOP.LOCK/ON,           ;READ SECOND HDR BYTE
;8416   ;       INCR.LPCNTR/ON,
;8417   ;       BENO/CNTR.OVFLW,
;8418   ;       NAD/READ.HDR.BYTE4A
;8419   ;
;8420   ;-----------------------------------------------------------------------------------
;8421   ;=1
;8422   ;
;8423   ;STORE.HDR.BYTE4A:
;8424   ;
;8425   ;       LOOP.LOCK/ON,           ;STORE SECOND HDR BYTE IN FIFO
;8426   ;       ENB.DSR/ON,
;8427   ;       WRT.FIFO/ON,
;8428   ;       INCR.FIFO.CNTR/ON,
;8429   ;       LOAD.LOOP.CNTR.[0F2]
;8430   ;
;8431   ;-----------------------------------------------------------------------------------
;8432   ;=0
;8433   ;
;8434   ;SKIP.HDR.WORD4:
;8435   ;
;8436   ;       LOOP.LOCK/ON,           ;SKIP THIRD & FOURTH HEADER BYTE
;8437   ;       INCR.LPCNTR/ON,
;8438   ;       BENO/CNTR.OVFLW,
;8439   ;       NAD/SKIP.HDR.WORD4
;8440   ;
;8441   ;-----------------------------------------------------------------------------------
;8442   ;=1
;8443   ;
;8444   ;SKIP.HDR.WORD4A:
;8445   ;
;8446   ;       LOOP.LOCK/ON,           ;SETUP FOR HDR CRC BYTES
;8447   ;       LOAD.LOOP.CNTR.[0F8]
;8448   ;
;8449   ;-----------------------------------------------------------------------------------
;8450   ;=0
;8451   ;
;8452   ;READ.CRC.BYTE4:
;8453   ;
```

```
;8454   ;       LOOP.LOCK/ON,               ;READ FIRST HEADER CRC BYTE
;8455   ;       INCR.LPCNTR/ON,
;8456   ;       BENO/CNTR.OVFLW,
;8457   ;       NAD/READ.CRC.BYTE4
;8458   ;
;8459   ;------------------------------------------------------------------------
;8460   ;=1
;8461   ;
;8462   ;STORE.CRC.BYTE4:
;8463   ;
;8464   ;       LOOP.LOCK/ON,               ;STORE FIRST HDR CRC BYTE IN FIFO
;8465   ;       ENB.DSR/ON,
;8466   ;       WRT.FIFO/ON,
;8467   ;       INCR.FIFO.CNTR/ON,
;8468   ;       LOAD.LOOP.CNTR.[0F9]
;8469   ;
;8470   ;------------------------------------------------------------------------
;8471   ;=0
;8472   ;
;8473   ;READ.CRC.BYTE4A:
;8474   ;
;8475   ;       LOOP.LOCK/ON,               ;READ SECOND HDR CRC BYTE
;8476   ;       INCR.LPCNTR/ON,
;8477   ;       BENO/CNTR.OVFLW,
;8478   ;       NAD/READ.CRC.BYTE4A
;8479   ;
;8480   ;------------------------------------------------------------------------
;8481   ;=1
;8482   ;
;8483   ;STORE.CRC.BYTE4A:
;8484   ;
;8485   ;       LOOP.LOCK/ON,               ;STORE SECOND HDR CRC BYTE IN FIFO
;8486   ;       ENB.DSR/ON,
;8487   ;       INCR.FIFO.CNTR/ON,
;8488   ;       WRT.FIFO/ON
;8489   ;
;8490   ;------------------------------------------------------------------------
;8491   ;
;8492   ;CLEAR.FIFO.ADRS4:
;8493   ;
;8494   ;       LOOP.LOCK/ON,
;8495   ;       CLR.FIFO.ADR,               ;CLEAR THE FIFO ADRS COUNTER.
;8496   ;       NAD/CHGCLK4B
;8497   ;
;8498   ;------------------------------------------------------------------------
;8499   ;
;8500   ;CHGCLK4B:
;8501   ;
;8502   ;       LOOP.LOCK/ON,               ;SWITCH OVER TO CPU CLOCK
;8503   ;       ENB.CPU.CLK                 ;DURING POSTAMBLE.
;8504   ;
;8505   ;------------------------------------------------------------------------
;8506   ;
;8507   ;XFER.DONE4:
;8508   ;
```

```
;8509   ;       UCMD/SET.CRDY,                  ;SIGNAL END OF TRANSFER.
;8510   ;       FUNC/SET.INT,
;8511   ;       NAD/SET.INT.AND.IDLE
;8512   ;
;8513   ;-----------------------------------------------------------------------------
;8514   ;=111111110
;8515   ;
;8516   ;SET.INT.AND.IDLE:
;8517   ;
;8518   ;       FUNC/SET.INT,                   ; The purpose of this state is to give CRDY one
;8519   ;       NAD/IDLE                        ; state to reset before asserting interrupt.
;8520   ;                                       ; CRDY is now used in the INT equation.
;8521   ;
;8522   ;-----------------------------------------------------------------------------
;8523   ;=1
;8524   ;
;8525   ;RTN.TO.IDLE:
;8526   ;IDLE:
;8527   ;
;8528   ;       BENO/CRDY.L,                    ;LOOK FOR GO (CRDY L)
;8529   ;       NAD/SETUP.DRV.CHK
;8530   ;-----------------------------------------------------------------------------
;8531   ;=0
;8532   ;
;8533   ;SETUP.DRV.CHK:
;8534   ;
;8535   ;       LOAD.LOOP.CNTR.[OFB],           ;COUNTER WILL BE USED
;8536   ;       NAD/SEL.NXT.DRV                 ;TO SEQUENCE THRU DRIVES.
;8537   ;
;8538   ;-----------------------------------------------------------------------------
;8539   ;
;8540   ;SEL.NXT.DRV:
;8541   ;
;8542   ;       SEL.DRV/UPROG,
;8543   ;       BENO/CNTR.OVFLW,
;8544   ;       NAD/CHK.DRV.STATUS
;8545   ;
;8546   ;-----------------------------------------------------------------------------
;8547   ;=0
;8548   ;
;8549   ;CHK.DRV.STATUS:
;8550   ;       ; Give drive status time to settle down.
;8551   ;
;8552   ;       SEL.DRV/UPROG,                  ;SELECT DRIVE
;8553   ;       NAD/DRV.STATUS.DISPATCH
;8554   ;
;8555   ;-----------------------------------------------------------------------------
;8556   ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
;8557   ;
;8558   ;DRV.STATUS.DISPATCH:
;8559   ;       SEL.DRV/UPROG,                  ;SELECT DRIVE
;8560   ;       BEN1/DRIVE.ONLINE,              ;AND BRANCH ON STATUS.
;8561   ;       BENO/DRIVE.RDY,
;8562   ;       NAD/-ERR,-OL,-RDY
;8563   ;
```

```
                  ;
                  ;-------------------------------------------------------------------
                  ;=011
                  ;
                  ;-ERR.OL.RDY:
                  ;
                  ;       SEL.DRV/UPROG,            ;NOP
                  ;       INCR.LPCNTR/ON,           ;ADD 1 TO DRIVE SELECT.
                  ;       NAD/SEL.NXT.DRV
                  ;
                  ;-------------------------------------------------------------------
                  ;
                  ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                  ;
                  ;              UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                  ;
                  ;-------------------------------------------------------------------
                  ;
                  ;=1
                  ;
                  ;RTN.TO.IDLE:
                  ;IDLE:
                  ;
                  ;       BENO/CRDY.L,              ;LOOK FOR GO (CRDY L)
                  ;       NAD/SETUP.DRV.CHK
                  ;
                  ;-------------------------------------------------------------------


                  ;Set up parameters for time-out error report :

U 0861, 36.4095           MOV LS[#1] TO WR[1]
U 0862, 3E.8295           MOV WR[1] TO LS[ERROR.NUMBER]          ;set up error number = 1 in LS
U 0863, E5.8A15           CLR LS[ERROR.MASK]                     ;no error mask used here
U 0864, 36.4A95           MOV LS[IDC] TO WR[1]                   ;store module code in LS to
U 0865, BE.8C95           MOV WR[1] TO LS[MODULE.NUM]            ; modules
U 0866, 36.6E95           MOV LS[NA.EXP.REC] TO WR[1]            ;put "N/A" for printout data
U 0867, BE.8095           MOV WR[1] TO LS[CONTROL.STATUS]

U 0868, 2F.8015           CLR WR[0]                              ;Initialize time-out counter
U 0869, BF.8C15           MOV WR[0] TO LS[TIMEOUT.CNTR]

                  WAIT.IDC.T5.1:
U 086A, 37.8C15           MOV LS[TIMEOUT.CNTR] TO WR[0]          ;load the current time count
U 086B, 37.9095           MOV LS[WAIT.TIME1] TO WR[1]            ;time limit on read header
                          CMP WR[0] WITH WR[1],                  ;timeout yet ???
U 086C, 26.4085            DT(LONG)&SET.ALU.CC
U 086D, 08.8739           JMP.IF[EQL] TO [T5.ERR.1]              ;yes, timeout on read header
U 086E, 20.4015           INC WR[0]                              ;inc. time count
U 086F, BF.8C15           MOV WR[0] TO LS[TIMEOUT.CNTR]
U 0870, 88.700C           JSR [WAIT.CSR.CRDY]                    ;check if CRDY is set
U 0871, 08.8784           JMP [IDC.DONE.T5.1]                    ;yes, CRDY is set now
U 0872, 08.86A4           JMP [WAIT.IDC.T5.1]                    ;no, wait some more

                  ;Force an error in the CHECK.RESULT routine for this timeout :
                        ;8619
                        ;8620   T5.ERR.1:
U 0873, 2F,8015         ;8621           CLR WR[0]                                   ;set unequal values
U 0874, 36,4095         ;8622           MOV LS[#1] TO WR[1]                         ; between WR0 and WR1
U 0875, 08,693C         ;8623           JSR [CHECK.RESULT]                          ;force an error
U 0876, 88,85E4         ;8624           JMP [T.5.2]                                 ;err and if LOE is set
U 0877, 08,8784         ;8625           JMP [IDC.DONE.T5.1]
                        ;8626
                        ;8627   ;Set up error parameters for ERROR 2 report :
                        ;8628
                        ;8629   IDC.DONE.T5.1:
U 0878, 36,4215         ;8630           MOV LS[#2] TO WR[0]                         ;error number = 2
U 0879, BE,8215         ;8631           MOV WR[0] TO LS[ERROR.NUMBER]
U 087A, B6,6E15         ;8632           MOV LS[NA.EXP.REC] TO WR[0]                 ;put "N/A" for printout data
U 087B, 3E,8015         ;8633           MOV WR[0] TO LS[CONTROL.STATUS]
U 087C, 2F,8015         ;8634           CLR WR[0]
U 087D, 3E,8A15         ;8635           MOV WR[0] TO LS[ERROR.MASK]                 ;no errror mask used
U 087E, B6,4A15         ;8636           MOV LS[IDC] TO WR[0]                        ;store IDC in LS as the
U 087F, 3E,8C15         ;8637           MOV WR[0] TO LS[MODULE.NUM]                 ;  module at fault
                        ;8638
                        ;8639   ;Read the header word from FIFO A.
                        ;8640   ;Position the heads to the innermost cylinder of the disk.
                        ;8641
                        ;8642   T.5.3:
U 0880, 88,72EC         ;8643           JSR [SHIFT.HDR]                             ;shift the header to DAR format
U 0881, B7,EC95         ;8644           MOV LS[#0000FF80] TO WR[1]                  ;value for seek cylinder
U 0882, BF,8495         ;8645           MOV WR[1] TO LS[SEEK.CYL]                   ;fetch cylinder to be sought
U 0883, 08,745C         ;8646           JSR [SEEK.DAR]                              ;setup the DAR word for seek
                        ;8647
U 0884, B6,0215         ;8648           MOV LS[SAV.WR0] TO WR[0]                    ;WR0 has the right DAR for seek
                        ;8649                                                       ; as the output from JSR [SEEK.DAR]
U 0885, 97,A415         ;8650           PORT.WRITE PORT.ADRS[DAR] WITH WR[0]        ;write to DAR with this format
                        ;8651
                        ;8652   T5.SEEK:
U 0886, B7,0495         ;8653           MOV LS[SEEK] TO WR[1]                       ;get the seek format
U 0887, C7,5095         ;8654           BIS LS[SETCSR.DS1] TO WR[1]                 ;select drive 1
U 0888, 97,A095         ;8655           PORT.WRITE PORT.ADRS[CSR] WITH WR[1]        ;seek to the desired cylinder
                        ;8656
                        ;8657
                        ;8658
                        ;8659   ;*** RL02 SEEK IDC FUNCTION CODES : ***
                        ;8660
                        ;8661   ;-------------------------------------------------------------------
                        ;8662   ;=1
                        ;8663   ;
                        ;8664   ;TEST.OR.RUN:
                        ;8665   ;
                        ;8666   ;       BEN2/MAINT,             ;IS THAT YOU, VICKIE?
                        ;8667   ;       NAD/DRIVE.TYPE
                        ;8668   ;
                        ;8669   ;-------------------------------------------------------------------
                        ;8670   ;=0**
                        ;8671   ;
                        ;8672   ;DRIVE.TYPE:
                        ;8673   ;
```

```
;       BEN0/R80,                   ;WHICH TYPE OF DRIVE WAS SELECTED?
;       NAD/RL02.IFORK
;
;-----------------------------------------------------------------------
;=0
;
;RL02.IFORK:
;
;       UCMD/SET.FIFOA,
;       FUNC/CLR.ECC.ERROR,         ; Enable clear of ECC/CRC errors for new function
;       BEN0/F0,                    ;8 WAY INSTRUCTION FORK.
;       BEN1/F1,                    ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
;       BEN2/F2,
;       NAD/NOP.CMD
;
;-----------------------------------------------------------------------
;=011
;
;SEEK.CMD:
;
;       NAD/WAIT.DRDY3              ;START OF THE RL02 SEEK COMMAND.
;
;-----------------------------------------------------------------------
;=0
;
;WAIT.DRDY3:
;
;       BEN0/DRIVE.RDY,             ;WAIT FOR DRIVE READY
;       NAD/WAIT.DRDY3
;
;-----------------------------------------------------------------------
;=1
;
;CHGCLK3:
;
;       ENB.SYS.CLK                 ;USE RL02 WRITE CLOCK
;
;-----------------------------------------------------------------------
;=0*
;
;WAIT.SEC.PLS3:
;
;       BEN1/SECTOR.PLS,            ;WAIT FOR LEADING EDGE OF SECTOR PULSE
;       NAD/WAIT.SEC.PLS3
;
;-----------------------------------------------------------------------
;=1*
;
;SEC.PLS.FND3:
;
;       NAD/WAIT.END.SEC.PLS3       ;SECTOR PULSE HAS BEEN FOUND
;
;-----------------------------------------------------------------------
;=0*
;
```

```
;END.SEC.PLS3:
;
;       LOAD.LOOP.CNTR.[2E],      ;PREPARE FOR WAIT LOOP
;       NAD/WAIT.50.US3
;
;-----------------------------------------------------------------
;=1*
;
;WAIT.END.SEC.PLS3:
;
;       BEN1/SECTOR.PLS,          ;WAIT FOR END OF SECTOR PULSE
;       NAD/WAIT.END.SEC.PLS3
;
;-----------------------------------------------------------------
;=0
;
;WAIT.50.US3:
;
;       INCR.LPCNTR/ON,           ;WAIT 50 US BEFORE SENDING DRIVE CMD.
;       ENB.CLR.BC/ON,            ;CLEAR BC SO THAT DARO IS SELECTED
;       BENO/CNTR.OVFLW,
;       NAD/WAIT.50.US3
;
;-----------------------------------------------------------------
;=1
;
;SETUP.DRV.CMD3:
;
;       ENB.CLR.BC/ON,            ;PREPARE TO SEND COMMAND.
;       LOAD.LOOP.CNTR.[0F0]
;
;-----------------------------------------------------------------
;=0
;
;SEND.DRV.CMD3:
;
;       ENB.DRV.CMD/ON,           ;SHIFT OUT DRIVE COMMAND
;       INCR.LPCNTR/ON,
;       BENO/CNTR.OVFLW,
;       NAD/SEND.DRV.CMD3
;
;-----------------------------------------------------------------
;=1
;
;DRV.CMD.DONE3:
;
;       ENB.CPU.CLK,              ;SWITCH BACK TO IDLE CLOCK
;       FUNC/CLR.ONLINE,          ; Avoid setting attn bit on -DRDY
;       NAD/XFER.DONE4
;
;-----------------------------------------------------------------
;
;XFER.DONE4:
;
;       UCMD/SET.CRDY,            ;SIGNAL END OF TRANSFER.
```

```
;8784  ;        FUNC/SET.INT,
;8785  ;        NAD/SET.INT.AND.IDLE
;8786  ;
;8787  ;----------------------------------------------------------------------
;8788  ;=111111110
;8789  ;
;8790  ;SET.INT.AND.IDLE:
;8791  ;
;8792  ;        FUNC/SET.INT,             ; The purpose of this state is to give CRDY one
;8793  ;        NAD/IDLE                  ; state to reset before asserting interrupt.
;8794  ;                                  ; CRDY is now used in the INT equation.
;8795  ;
;8796  ;----------------------------------------------------------------------
;8797  ;=1
;8798  ;
;8799  ;RTN.TO.IDLE:
;8800  ;IDLE:
;8801  ;
;8802  ;        BENO/CRDY.L,              ;LOOK FOR GO (CRDY L)
;8803  ;        NAD/SETUP.DRV.CHK
;8804  ;----------------------------------------------------------------------
;8805  ;=0
;8806  ;
;8807  ;SETUP.DRV.CHK:
;8808  ;
;8809  ;        LOAD.LOOP.CNTR.[OFB],     ;COUNTER WILL BE USED
;8810  ;        NAD/SEL.NXT.DRV           ;TO SEQUENCE THRU DRIVES.
;8811  ;
;8812  ;----------------------------------------------------------------------
;8813  ;
;8814  ;SEL.NXT.DRV:
;8815  ;
;8816  ;        SEL.DRV/UPROG,
;8817  ;        BENO/CNTR.OVFLW,
;8818  ;        NAD/CHK.DRV.STATUS
;8819  ;
;8820  ;----------------------------------------------------------------------
;8821  ;=0
;8822  ;
;8823  ;CHK.DRV.STATUS:
;8824  ;        ; Give drive status time to settle down.
;8825  ;
;8826  ;        SEL.DRV/UPROG,            ;SELECT DRIVE
;8827  ;        NAD/DRV.STATUS.DISPATCH
;8828  ;
;8829  ;----------------------------------------------------------------------
;8830  ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
;8831  ;
;8832  ;DRV.STATUS.DISPATCH:
;8833  ;        SEL.DRV/UPROG,            ;SELECT DRIVE
;8834  ;        BEN1/DRIVE.ONLINE,        ;AND BRANCH ON STATUS.
;8835  ;        BENO/DRIVE.RDY,
;8836  ;        NAD/-ERR.-OL.-RDY
;8837  ;
;8838  ;
```

```
                  ;8839   ;-------------------------------------------------------------------------
                  ;8840   ;=011
                  ;8841   ;
                  ;8842   ;-ERR.OL.RDY:
                  ;8843   ;
                  ;8844   ;       SEL.DRV/UPROG,          ;NOP
                  ;8845   ;       INCR.LPCNTR/ON,         ;ADD 1 TO DRIVE SELECT.
                  ;8846   ;       NAD/SEL.NXT.DRV
                  ;8847   ;
                  ;8848   ;-------------------------------------------------------------------------
                  ;8849   ;
                  ;8850   ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                  ;8851   ;
                  ;8852   ;                 UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                  ;8853   ;
                  ;8854   ;-------------------------------------------------------------------------
                  ;8855   ;
                  ;8856   ;=1
                  ;8857   ;
                  ;8858   ;RTN.TO.IDLE:
                  ;8859   ;IDLE:
                  ;8860   ;
                  ;8861   ;       BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
                  ;8862   ;       NAD/SETUP.DRV.CHK
                  ;8863   ;
                  ;8864   ;-------------------------------------------------------------------------
                  ;8865
U 0889, 2F,8015   ;8866           CLR WR[0]                               ;Initialize time-out counter
U 088A, BF,8C15   ;8867           MOV WR[0] TO LS[TIMEOUT.CNTR]
                  ;8868
                  ;8869   WAIT.IDC.T5.2:
U 088B, 37,8C15   ;8870           MOV LS[TIMEOUT.CNTR] TO WR[0]           ;load the current time count
U 088C, 37,9095   ;8871           MOV LS[WAIT.TIME1] TO WR[1]             ;time limit on seek command
                  ;8872           CMP WR[0] WITH WR[1],                   ;timeout yet ???
U 088D, 26,4085   ;8873            DT(LONG)&SET.ALU.CC
U 088E, 08,8949   ;8874           JMP.IF[EQL] TO [T5.ERR.2]               ;yes, timeout on seek command
U 088F, 20,4015   ;8875           INC WR[0]                               ;inc. time count
U 0890, BF,8C15   ;8876           MOV WR[0] TO LS[TIMEOUT.CNTR]
U 0891, 88,700C   ;8877           JSR [WAIT.CSR.CRDY]                     ;check if CRDY is set
U 0892, 08,8994   ;8878           JMP [T.5.4]                             ;yes, CRDY is set now
U 0893, 08,8884   ;8879           JMP [WAIT.IDC.T5.2]                     ;no, wait some more
                  ;8880
                  ;8881   ;Force an error in the CHECK.RESULT routine for this timeout :
                  ;8882
                  ;8883   T5.ERR.2:
U 0894, 2F,8015   ;8884           CLR WR[0]                               ;set unequal'values
U 0895, 36,4095   ;8885           MOV LS[#1] TO WR[1]                     ; between WR0 and WR1
U 0896, 08,693C   ;8886           JSR [CHECK.RESULT]                      ;force an error
U 0897, 88,8864   ;8887           JMP [T5.SEEK]                           ;err and if LOE is set
U 0898, 08,8994   ;8888           JMP [T.5.4]
                  ;8889
                  ;8890   ;Do another read header command to verify if the heads are really on the innermost
                  ;8891   ; cylinder :
                  ;8892
                  ;8893   T.5.4:
```

```
U 0899, B7,0615                 MOV LS[READ.HEADER] TO WR[0]            ;get the read header command
U 089A, 47,5015                 BIS LS[SETCSR.DS1] TO WR[0]             ;select drive 1
U 089B, 17,A015                 PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;write the CSR

                                ;*****************************************************************
                                ;* REFER TO THE RL02 READ HEADER IDC FUNCTION CODES AS LISTED ABOVE *
                                ;*****************************************************************

                        WAIT.IDC.T5.3:
U 089C, 88,700C                 JSR [WAIT.CSR.CRDY]                     ;check if CRDY is set
U 089D, 08,89F4                 JMP [IDC.DONE.T5.2]                     ;CRDY set !!!!
U 089E, 08,89C4                 JMP [WAIT.IDC.T5.3]                     ;wait some more

                        IDC.DONE.T5.2:
U 089F, 36,C015                 MOV LS[#3] TO WR[0]                     ;error number = 3
U 08A0, BE,8215                 MOV WR[0] TO LS[ERROR.NUMBER]
U 08A1, 37,A215                 MOV LS[#FFFF007F] TO WR[0]
U 08A2, 3E,8A15                 MOV WR[0] TO LS[ERROR.MASK]             ;no errror mask used
U 08A3, B6,4C15                 MOV LS[RL02] TO WR[0]                   ;store RL02 in LS as
U 08A4, 3E,8C15                 MOV WR[0] TO LS[MODULE.NUM]             ; the module at fault
U 08A5, E5,8015                 CLR LS[CONTROL.STATUS]                  ;clear [NA.EXP.REC] bit

                        ;Now read the header from FIFO A and shift that into the DAR format :

U 08A6, 88,72EC                 JSR [SHIFT.HDR]                         ;LS[SAV.WRO] has the received
                                                                        ; cylinder adr. in DAR format
U 08A7, B6,0215                 MOV LS[SAV.WRO] TO WR[0]                ;set up the two WR's for the
U 08A8, B7,EC95                 MOV LS[#0000FF80] TO WR[1]              ; JSR [CHECK.RESULT] routine
U 08A9, 08,693C                 JSR [CHECK.RESULT]                      ;verify the result
U 08AA, 88,8864                 JMP [T5.SEEK]                           ;err and if LOE is set
U 08AB, 08,8AC4                 JMP [T.5.5]                             ;test the outermost cylinder

                        ;Position the heads to the outermost cylinder of the disk.

                        T.5.5:
U 08AC, 2F,8295                 CLR WR[1]                               ;init. value for seek cylinder
U 08AD, BF,8495                 MOV WR[1] TO LS[SEEK.CYL]               ;fetch cylinder to be sought
U 08AE, 08,745C                 JSR [SEEK.DAR]                          ;setup the DAR seek word
U 08AF, 08,74FC                 JSR [SEEK.VERIFY]                       ;go seek and read header
U 08B0, B6,4495                 MOV LS[#4] TO WR[1]
U 08B1, 3E,8295                 MOV WR[1] TO LS[ERROR.NUMBER]           ;error = 4
U 08B2, B6,0215                 MOV LS[SAV.WRO] TO WR[0]                ;current cyl. where heads are
U 08B3, 37,8495                 MOV LS[SEEK.CYL] TO WR[1]               ;fetch the expected cylinder
U 08B4, 08,693C                 JSR [CHECK.RESULT]                      ;verify the received cyl. adr.
U 08B5, 08,8AC4                 JMP [T.5.5]                             ;loop on err if err bit set
U 08B6, 08,8B74                 JMP [END.T5]                            ;test ends

                        END.T5:                                         ;end of test 5

```

```
.PAGE  "Test 6  RL READ DATA TEST (look for "SCRATCH" label, not in CRD mode, optional in APT mode)"
;++
;
; Test Description:
;
;       This test will first look at LS[APT.PRESENT] and LS[APT.PARAM] to
;       see if it is under APT mode and if software register in APT has
;       chosen to run this test. And then it will check the READ DATA
;       function by reading the label in the home_block resident on cylinder
;       zero, sector 4. Read header and seek functions as described in the
;       "RL READ HEADER AND SEEK FUNCTION TEST" will be issued so as to
;       position the R/W heads to cylinder zero. The test will then write
;       to the DAR with the proper disk address format <bits 0 to 15>,
;       sector 4, head 0 and cylinder 0 in this case. Then the CSR will be
;       written with drive select 1 <bit 8> and the read data function
;       <bits 2 and 3>. The label name located from the 216th to the 227th
;       bytes of the 256 home_block bytes will be read from FIFO A and
;       compared with the expected data in the local store.
;
; Note : This test also verifies Read Data w/o header check function since
;        the microcode for this function is a subset of that used in the
;        Read Data function.
;
; Assumption:
;
;       Both CPU micros and IDC part I has been run successfully and disk
;       drive 1 is up and spinning and ready to accept commands. Both the
;       seek and read header functions are working properly.
;
;       This test assumes that homeblock is not a bad block.
;
; Test Steps:
;
;         1. Check if running APT from LS[APT.PRESENT], if so
;         2.     Then is APT software byte set for this test, if so
;         3.          Then go to step 6
;         4.          Else go to end of this test
;         5.     Else proceed
;         6. Write CSR with drive 1 select and read header function bits
;         7. Wait for CRDY to set
;         8. Port select FIFO A and clear its address counter
;         9. Read header from FIFO A
;        10. Mask the header with LS[#FFFFFF80] to get the cylinder address
;        11. Write DAR with marker and dir. bit set and the cylinder
;             address difference (from current cyl. adr. to cyl. 0)
;        12. Write CSR with drive 1 select and seek function bits
;        13. Wait for CRDY to set
;        14. Load WR[1] with the DAR format of sector 3, track 0 & cyl. 0
;        15. Write DAR with WR[1]
;        16. Write CSR with drive 1 select and read data function bits
;        17. Set up a timeout loop to wait for XFER REQ set
;        18. If timeout on this read data command;
;        19.     Then setup error parameters and report ERROR 1
;        20.     Else send TRANFER GRANT to clear the XFER REQ
;        19. End if; port select FIFO A and clear its address counter
```

```
                  ;         20. Step up FIFO A address to get to the 12-byte label and
                  ;              read it into LS locations for comparison.
                  ;         21. Get the label from the LS locations and compare that with
                  ;              the hex equivalent (53435241544348) for "SCRATCH" label
                  ;         22. If data not compare
                  ;         23.     Then set up error parameters and report ERROR 2
                  ;         24.     Else
                  ;         25. Test ends
                  ;
                  ; Error:
                  ;
                  ;  ERROR 1 - Timeout error waiting for XFER REQ after a read data command
                  ;
                  ;                 Printout:
                  ;                         EXPECTED      RECEIVED        MODULE
                  ;                         N/A           N/A             IDC
                  ;
                  ;  ERROR 2 - The label of the disk pack on drive 1 is not "SCRATCH"
                  ;
                  ;                        ****  WARNING  ****
                  ;
                  ;      Once this error is hit, entering "CO" upon micmon prompt will still default
                  ;      in loop-on-error as a precaution from erasing data on the disk in subsequent
                  ;      tests. To intentionally continue one has to start from test 7.
                  ;
                  ;                 Printout:
                  ;                         EXPECTED      RECEIVED        MODULE
                  ;                         pack label    pack label      IDC
                  ;                         in ASCII      in ASCII
                  ;
                  ; Debug:
                  ;
                  ;  ERROR 1: Check RL DRIVE READY, RL SYSTEM CLOCK, SYNC SEEN, DS CLOCK, RL
                  ;           SECTOR PLS, RL SEC PLS & RL READ DATA signals which might have
                  ;           triggered the timeout if at fault.
                  ;
                  ;  ERROR 2: A pack with the label "SCRATCH" is needed in drive 1.
                  ;
                  ;--
                  T.6:
U 08B7, B6,5E15           MOV LS[BEGIN.TEST] TO WR[0]              ;set bit 15 in WR[0] for
                                                                    ; control and status word
U 08B8, 3E,8015           MOV WR[0] TO LS[CONTROL.STATUS]          ;set bit 15 in control and
                                                                    ; status word - bit 15
                                                                    ; indicates start of test to
                                                                    ; console
U 08B9, 10,E015           MISC [SET.CP.ATTN]                        ;issue CPU ATTN to console

                  WAIT.T6.0:
U 08BA, 88,8BA4           JMP [WAIT.T6.0]                           ;loop here waiting for
                                                                    ; console to respond

                  ;Store the LS[LABEL.SAV'S] here so that error report will print out ASCII instead
                  ; of hex values :
```

```
U 08BB, B7,2815   ;9053            MOV LS[#53435241] TO WR[0]            ;hex value for "SCRA"
U 08BC, 3F,1415   ;9054            MOV WR[0] TO LS[LABEL.SAV1]           ; to be printed upon error
U 08BD, 37,2A15   ;9055            MOV LS[#54434820] TO WR[0]            ;hex value for "TCH "
U 08BE, BF,1615   ;9056            MOV WR[0] TO LS[LABEL.SAV2]           ; to be printed upon error
U 08BF, 37,2C15   ;9057            MOV LS[#20202020] TO WR[0]            ;hex value for "    "
U 08C0, 3F,1815   ;9058            MOV WR[0] TO LS[LABEL.SAV3]           ; to be printed upon error
                  ;9059
U 08C1, 65,A015   ;9060            CLR LS[PREVIOUS.ERROR]                ;clear previous error number
                  ;9061                                                  ; in LS
U 08C2, 17,CC15   ;9062            PORT.CONTROL [CLEAR.IDC]              ;init. IDC
U 08C3, 17,CC15   ;9063            PORT.CONTROL [CLEAR.IDC]
U 08C4, 17,CC15   ;9064            PORT.CONTROL [CLEAR.IDC]
                  ;9065
                  ;9066  T.6.1:
U 08C5, 36,9015   ;9067            MOV LS[ERROR.CONTROL] TO WR[0]        ;look at the error control word
                  ;9068            BIT LS[APT.PRESENT] WITH WR[0],       ;see if it is running APT
U 08C6, 59,4A35   ;9069             DT(LONG)&SET.ALU.CC
U 08C7, 08,8CE9   ;9070            JMP.IF[BITS.CLR] TO [T.6.2]           ;no, go on with the test
                  ;9071
                  ;9072  ;Running APT, this will check if the software byte of the APT run time parameter register
                  ;9073  ; is set up to select to run this test :
                  ;9074
U 08C8, B6,C815   ;9075            MOV LS[#00FF00FF] TO WR[0]
U 08C9, C2,2C15   ;9076            AND LS[#FFFFFF00] TO WR[0]            ;create a mask for the software byte
U 08CA, 36,9295   ;9077            MOV LS[APT.PARAM] TO WR[1]            ;look at the APT run time register
                  ;9078            AND WR[0] TO WR[1],                   ;mask out the software byte
U 08CB, AF,C0B5   ;9079             DT(LONG)&SET.ALU.CC                  ;did APT select this test ??
U 08CC, 08,8CE9   ;9080            JMP.IF[EQL] TO [T.6.2]                ;yes, go run the test !!!!
U 08CD, 08,9324   ;9081            JMP [END.T6]                          ;no, skip this test.
                  ;9082
                  ;9083  ;Now, wherever the heads are, they have to go back to the outermost cylinder to read
                  ;9084  ; the pack label from the home-block. To accomplish this, similar codes from test 5
                  ;9085  ; are used here :
                  ;9086
                  ;9087  T.6.2:
U 08CE, B7,0615   ;9088            MOV LS[READ.HEADER] TO WR[0]          ;get the read header command
U 08CF, 47,5015   ;9089            BIS LS[SETCSR.DS1] TO WR[0].          ;select drive 1
U 08D0, 17,A015   ;9090            PORT.WRITE PORT.ADRS[CSR] WITH WR[0]  ;write the CSR
                  ;9091
                  ;9092            ;***********************************************************
                  ;9093            ;* REFER TO THE RL02 READ HEADER IDC FUNCTION CODES AS LISTED ABOVE *
                  ;9094            ;***********************************************************
                  ;9095
                  ;9096  WAIT.IDC.T6.1:
U 08D1, 88,700C   ;9097            JSR [WAIT.CSR.CRDY]                   ;check if CRDY is set
U 08D2, 08,8D44   ;9098            JMP [IDC.DONE.T6.1]                   ;CRDY set !!!!
U 08D3, 08,8D14   ;9099            JMP [WAIT.IDC.T6.1]                   ;wait some more
                  ;9100
                  ;9101  ;Get the desired seek cylinder and read the header, to get the cylinder difference:
                  ;9102
                  ;9103  IDC.DONE.T6.1:
U 08D4, 2F,8015   ;9104            CLR WR[0]
U 08D5, 3F,8415   ;9105            MOV WR[0] TO LS[SEEK.CYL]
U 08D6, 88,72EC   ;9106            JSR [SHIFT.HDR]
                  ;9107
                  ;9108  ;Put that in a seekable DAR format which is stored at LS[SAV.WR0] upon completion
                  ;9109  ; of the following routine :
                  ;9110
U 08D7, 08,745C   ;9111          JSR [SEEK.DAR]
                  ;9112
                  ;9113  ;Do a seek to the desired outermost cylinder :
                  ;9114
U 08D8, B6,0215   ;9115          MOV LS[SAV.WR0] TO WR[0]                   ;get the DAR format and
U 08D9, 97,A415   ;9116          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]      ; write to the DAR
                  ;9117
U 08DA, 37,0415   ;9118          MOV LS[SEEK] TO WR[0]                      ;get the seek command and
U 08DB, 47,5015   ;9119          BIS LS[SETCSR.DS1] TO WR[0]               ; select drive 1
U 08DC, 17,A015   ;9120          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;do a seek to the drive
                  ;9121
                  ;9122          ;*****************************************************************
                  ;9123          ;* REFER TO THE RL02 SEEK IDC FUNCTION CODES AS LISTED IN TEST 5 *
                  ;9124          ;*****************************************************************
                  ;9125
                  ;9126  WAIT.IDC.T6.2:
U 08DD, 88,700C   ;9127          JSR [WAIT.CSR.CRDY]                        ;check if CRDY is set
U 08DE, 88,8E34   ;9128          JMP [T.6.3]                                ;CRDY set !!!!
U 08DF, 08,8DD4   ;9129          JMP [WAIT.IDC.T6.2]                        ;wait some more
                  ;9130
                  ;9131  ;See if the RL02 is ready to take the next operation :
                  ;9132
                  ;9133  CHECK.DRDY.T6.1:
U 08E0, 08,708C   ;9134          JSR [WAIT.CSR.DRDY]                        ;wait for DRDY to set
                  ;9135
U 08E1, 17,D815   ;9136          PORT.CONTROL [SEL.FIFO.A]                  ;port select FIFO A
U 08E2, 17,C015   ;9137          PORT.CONTROL [CLEAR.FIFO.CNTR]             ;empty out FIFO A
                  ;9138
                  ;9139  ;Now it's time to read the home_block sector the label should be in the
                  ;9140  ; 4th sector of cylinder 0 if there was no prior bad block :
                  ;9141
                  ;9142  T.6.3:
U 08E3, 36,C015   ;9143          MOV LS[#3] TO WR[0]                        ;get to the 4th sector
U 08E4, 97,A415   ;9144          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]      ; with the DAR read format
                  ;9145
U 08E5, B7,0A15   ;9146          MOV LS[READ.DATA] TO WR[0]                 ;set up a read data command
U 08E6, 47,5015   ;9147          BIS LS[SETCSR.DS1] TO WR[0]               ;select drive 1
U 08E7, 17,A015   ;9148          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;write the CSR
                  ;9149
                  ;9150  ;*** RL02 READ DATA IDC FUNCTION CODES : ***
                  ;9151
                  ;9152  ;-----------------------------------------------------------------------
                  ;9153  ;=1
                  ;9154  ;
                  ;9155  ;TEST.OR.RUN:
                  ;9156  ;
                  ;9157  ;       BEN2/MAINT,               ;IS THAT YOU, VICKIE?
                  ;9158  ;       NAD/DRIVE.TYPE
                  ;9159  ;
                  ;9160  ;-----------------------------------------------------------------------
                  ;9161  ;=0**
                  ;9162  ;
```

```
;9163  ;DRIVE.TYPE:
;9164  ;
;9165  ;       BENO/R80,                  ;WHICH TYPE OF DRIVE WAS SELECTED?
;9166  ;       NAD/RL02.IFORK
;9167  ;
;9168  ;------------------------------------------------------------------------
;9169  ;=0
;9170  ;
;9171  ;RL02.IFORK:
;9172  ;
;9173  ;       UCMD/SET.FIFOA,
;9174  ;       FUNC/CLR.ECC.ERROR,        ; Enable clear of ECC/CRC errors for new function
;9175  ;       BENO/F0,                   ;8 WAY INSTRUCTION FORK.
;9176  ;       BEN1/F1,                   ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
;9177  ;       BEN2/F2,
;9178  ;       NAD/NOP.CMD
;9179  ;
;9180  ;------------------------------------------------------------------------
;9181  ;=110
;9182  ;
;9183  ;RD.DAT.CMD:
;9184  ;
;9185  ;       SET.FIFO.ADR,              ;START OF THE RL02 READ DATA COMMAND.
;9186  ;       NAD/INIT.FIFO6             ;PRESET FIFOA ADRS CNTR TO OVFLW.
;9187  ;
;9188  ;------------------------------------------------------------------------
;9189  ;
;9190  ;INIT.FIFO6:
;9191  ;
;9192  ;       UCMD/CHG.FIFO              ;CHANGE TO FIFOB
;9193  ;
;9194  ;------------------------------------------------------------------------
;9195  ;
;9196  ;INIT.FIFO6A:
;9197  ;
;9198  ;       SET.FIFO.ADR               ;PRESET FIFOB ADRS CNTR TO OVFLW.
;9199  ;
;9200  ;------------------------------------------------------------------------
;9201  ;
;9202  ;INIT.FIFO6B:
;9203  ;
;9204  ;       UCMD/SET.FIFOA,            ;RETURN TO FIFOA
;9205  ;       NAD/WAIT.DRDY1
;9206  ;
;9207  ;------------------------------------------------------------------------
;9208  ;=0
;9209  ;
;9210  ;WAIT.DRDY1:
;9211  ;
;9212  ;       BENO/DRIVE.RDY,            ;WAIT FOR DRIVE READY
;9213  ;       NAD/WAIT.DRDY1
;9214  ;
;9215  ;------------------------------------------------------------------------
;9216  ;=1
;9217  ;
```

```
;9218   ;CHGCLK1:
;9219   ;
;9220   ;       ENB.SYS.CLK                 ;USE RL02 WRITE CLOCK
;9221   ;
;9222   ;------------------------------------------------------------------------
;9223   ;=0*
;9224   ;
;9225   ;WAIT.SEC.PLS1:
;9226   ;
;9227   ;       BEN1/SECTOR.PLS,            ;WAIT FOR LEADING EDGE OF SECTOR PULSE
;9228   ;       NAD/WAIT.SEC.PLS1
;9229   ;
;9230   ;------------------------------------------------------------------------
;9231   ;=1*
;9232   ;
;9233   ;SEC.PLS.FND1:
;9234   ;
;9235   ;       NAD/WAIT.END.SEC.PLS1       ;SECTOR PULSE HAS BEEN FOUND
;9236   ;
;9237   ;------------------------------------------------------------------------
;9238   ;=1*
;9239   ;
;9240   ;WAIT.END.SEC.PLS1:
;9241   ;
;9242   ;       BEN1/SECTOR.PLS,            ;WAIT FOR END OF SECTOR PULSE
;9243   ;       NAD/WAIT.END.SEC.PLS1
;9244   ;
;9245   ;------------------------------------------------------------------------
;9246   ;=1
;9247   ;
;9248   ;SETUP.RD.HDR1:
;9249   ;
;9250   ;       LOOP.LOCK/ON,               ;SETUP INITIAL CONDITIONS
;9251   ;       ENB.CLR.BC/ON,              ;FOR READING HEADER.
;9252   ;       FUNC/CLR.DSR,
;9253   ;       LOAD.LOOP.CNTR.[0F1]        ;256-15=241=F1
;9254   ;
;9255   ;------------------------------------------------------------------------
;9256   ;
;9257   ;CHGCLK1A:
;9258   ;
;9259   ;       ENB.DSK.CLK,                ;THE MICROSEQUENCER WILL STALL
;9260   ;       LOOP.LOCK/ON,               ;UNTIL SYNC SEEN IS SET (SYNC BIT FOUND)
;9261   ;       ENB.CLR.BC/ON,              ;THEN WE USE CLOCK FROM DATA SEPARATOR.
;9262   ;       GEN.SYNC.[80]
;9263   ;
;9264   ;------------------------------------------------------------------------
;9265   ;=0
;9266   ;
;9267   ;READ.HDR1:
;9268   ;
;9269   ;       LOOP.LOCK/ON,               ;SHIFT IN 15 MORE HEADER BITS AND
;9270   ;       DATA.FIELD/ON,              ;COMPARE WITH SERIAL DAR.
;9271   ;       ENB.CLR.BC/ON,
;9272   ;       INCR.LPCNTR/ON,
;9273   ;       BENO/CNTR.OVFLW,
;9274   ;       NAD/READ.HDR1
;9275   ;
;9276   ;---------------------------------------------------------------------
;9277   ;=1
;9278   ;
;9279   ;TEST.HDR.MATCH1:
;9280   ;
;9281   ;       LOOP.LOCK/ON,           ;HEADER HAS BEEN READ.
;9282   ;       UCON/TEST.MATCH,        ;TURN ON TEST MATCH, OFF ENB CLR BC.
;9283   ;       DATA.FIELD/ON,
;9284   ;       NAD/BRANCH.HDR.MATCH1
;9285   ;
;9286   ;---------------------------------------------------------------------
;9287   ;
;9288   ;BRANCH.HDR.MATCH1:
;9289   ;
;9290   ;       LOOP.LOCK/ON,
;9291   ;       DATA.FIELD/ON,          ;RESULTS OF HDR COMP NOW IN MISMATCH.
;9292   ;       BEN2/MISMATCH.L,
;9293   ;       NAD/HDR.COMP1
;9294   ;
;9295   ;---------------------------------------------------------------------
;9296   ;=1**
;9297   ;
;9298   ;HDR.COMP1:
;9299   ;
;9300   ;       LOOP.LOCK/ON,           ;HEADER COMPARE!
;9301   ;       DATA.FIELD/ON,          ;CONTINUE TO READ THE HDR DENSITY WORD.
;9302   ;       LOAD.LOOP.CNTR.[0F2],   ;256-14=242=F2
;9303   ;       NAD/RD.DENSITY.WD1
;9304   ;
;9305   ;---------------------------------------------------------------------
;9306   ;=0
;9307   ;
;9308   ;RD.DENSITY.WD1:
;9309   ;
;9310   ;       INCR.LPCNTR/ON,         ;CONT TO SHIFT IN THE HDR DENSITY WD.
;9311   ;       LOOP.LOCK/ON,
;9312   ;       DATA.FIELD/ON,
;9313   ;       BENO/CNTR.OVFLW,
;9314   ;       NAD/RD.DENSITY.WD1
;9315   ;
;9316   ;---------------------------------------------------------------------
;9317   ;=1
;9318   ;
;9319   ;CRC.PREP1:
;9320   ;
;9321   ;       LOAD.LOOP.CNTR.[0F1],   ;PREPARE TO READ THE CRC WORD.
;9322   ;       LOOP.LOCK/ON,           ;256-15=241=F1
;9323   ;       CHECK.FIELD/ON          ;TURN DFLD OFF AND CFLD ON.
;9324   ;
;9325   ;---------------------------------------------------------------------
;9326   ;=0
;9327   ;
```

```
;9328   ;RD.CRC.WD1:
;9329   ;
;9330   ;       INCR.LPCNTR/ON,             ;NOW WE SHIFT IN THE CRC WORD
;9331   ;       LOOP.LOCK/ON,
;9332   ;       CHECK.FIELD/ON,
;9333   ;       BENO/CNTR.OVFLW,
;9334   ;       NAD/RD.CRC.WD1
;9335   ;
;9336   ;-----------------------------------------------------------------------
;9337   ;=1
;9338   ;
;9339   ;CHECK.CRC1:
;9340   ;
;9341   ;       LOOP.LOCK/ON,
;9342   ;       UCON/CHECK.ECC              ;DROP CFLD. CLOCK ECC FLOP
;9343   ;
;9344   ;-----------------------------------------------------------------------
;9345   ;
;9346   ;TEST.FOR.ERR1:
;9347   ;
;9348   ;       LOOP.LOCK/ON,               ;WAS THERE A CRC ERROR?
;9349   ;       BEN1/ERROR.L
;9350   ;
;9351   ;-----------------------------------------------------------------------
;9352   ;=0*
;9353   ;
;9354   ;CRC.ERR1:
;9355   ;
;9356   ;       LOOP.LOCK/ON,
;9357   ;       ENB.CPU.CLK,                ;CRC ERROR. TERMINATE AND SET OPI.
;9358   ;       UCMD/SET.OPI,
;9359   ;       NAD/XFER.DONE1
;9360   ;
;9361   ;-----------------------------------------------------------------------
;9362   ;=1*
;9363   ;
;9364   ;NOT.CRC.ERR1:
;9365   ;
;9366   ;       LOOP.LOCK/ON,
;9367   ;       FUNC/CLR.ECC,               ;CLEAR DC007
;9368   ;       BEN2/F2,                    ;BRANCH ON FUNCTION BITS INTO WRT CHK,
;9369   ;       BEN1/F1,                    ;RD DATA OR WRT DATA FLOWS.
;9370   ;       NAD/BRANCH.WC.CMD1
;9371   ;
;9372   ;-----------------------------------------------------------------------
;9373   ;=11*
;9374   ;
;9375   ;BRANCH.RD.CMD6:
;9376   ;
;9377   ;       LOOP.LOCK/ON,               ;CONTINUE VIA READ DATA FLOWS.
;9378   ;       BEN1/FIFO.OVFLW.L,
;9379   ;       NAD/CHGCLK6B
;9380   ;
;9381   ;-----------------------------------------------------------------------
;9382   ;=0*
```

```
;
;CHGCLK6B:
;
;       LOOP.LOCK/ON,               ;EVERYTHING IS GO FOR READING DATA FROM
;       ENB.SYS.CLK,                ;THIS SECTOR. RUN OFF SYSTEM CLOCK THRU
;       LOAD.LOOP.CNTR.[0F5],       ;HEADER POSTAMBLE AND PAST SPLICE.
;       NAD/HDR.POST6
;
;-----------------------------------------------------------------------
;=0
;
;HDR.POST6:
;
;       INCR.LPCNTR/ON,             ;SHIFT OVER HEADER POSTAMBLE
;       BENO/CNTR.OVFLW,            ;LOOP LOCK OFF
;       NAD/HDR.POST6
;
;-----------------------------------------------------------------------
;=1
;
;DATA.PRE6:
;
;       FUNC/CLR.ECC,
;       LOOP.LOCK/ON,               ;NOW WE ARE IN THE DATA PREAMBLE
;       LOAD.LOOP.CNTR.[0E0]
;
;-----------------------------------------------------------------------
;=0
;
;DATA.PRE6A:
;
;       LOOP.LOCK/ON,               ;ALLOW PHASE LOCK LOOP 4 BYTES
;       INCR.LPCNTR/ON,             ;OF PREAMBLE TO LOCK UP
;       BENO/CNTR.OVFLW,
;       NAD/DATA.PRE6A
;
;-----------------------------------------------------------------------
;=1
;
;SETUP.RD.DATA6:
;
;       LOOP.LOCK/ON,               ;SETUP INITIAL CONDITIONS
;       CLR.FIFO.ADR,               ;FOR READING DATA.
;       LOAD.LOOP.CNTR.[0F9]
;
;-----------------------------------------------------------------------
;
;CHGCLK6C:
;
;       LOOP.LOCK/ON,               ;THE MICROSEQUENCER WILL STALL UNTIL
;       ENB.DSK.CLK,                ;SYNC SEEN IS SET.
;       GEN.SYNC.[80],              ;THEN WE USE THE DATA SEPARATOR CLK.
;       NAD/READ.DATA6
;
;-----------------------------------------------------------------------
```

```
;=10
;
;READ.DATA6:
;
;       LOOP.LOCK/ON,           ;SHIFT IN 7 DATA BITS
;       DATA.FIELD/ON,
;       INCR.LPCNTR/ON,
;       BEN0/CNTR.OVFLW,
;       NAD/READ.DATA6
;
;-----------------------------------------------------------------------
;=11
;
;STORE.BYTE6:
;
;       LOOP.LOCK/ON,           ;8TH DATA BIT
;       DATA.FIELD/ON,
;       ENB.DSR/ON,
;       WRT.FIFO/ON,            ;LOAD BYTE INTO FIFO
;       LOAD.LOOP.CNTR.[0F9],
;       INCR.FIFO.CNTR/ON,
;       BEN1/FIFO.MAX.L,
;       NAD/READ.DATA6
;
;-----------------------------------------------------------------------
;=00
;
;RD.DATA.CRC6:
;
;       LOOP.LOCK/ON,           ;START TO READ THE CRC WORD.
;       CHECK.FIELD/ON,         ;DROP DFLD, ASSERT CFLD
;       LOAD.LOOP/CNTR.[0F1],
;       NAD/RD.DATA.CRC6A
;
;-----------------------------------------------------------------------
;=0
;
;RD.DATA.CRC6A:
;
;       LOOP.LOCK/ON,
;       CHECK.FIELD/ON,
;       INCR.LPCNTR/ON,         ;SHIFT IN THE CRC WORD
;       BEN0/CNTR.OVFLW,
;       NAD/RD.DATA.CRC6A
;
;-----------------------------------------------------------------------
;=1
;
;CHECK.CRC6A:
;
;       LOOP.LOCK/ON,           ;DROP CFLD. CLOCK ECC FLOP
;       UCON/CHECK.ECC
;
;-----------------------------------------------------------------------
;
```

```
;9493  ;TEST.FOR.ERR6A:
;9494  ;
;9495  ;        LOOP.LOCK/ON,             ;SWITCH OVER TO CPU CLOCK
;9496  ;        ENB.CPU.CLK,              ;FOR FAST INTR OR TERMINATE.
;9497  ;        BEN1/ERROR.L              ;WAS THERE A CRC ERROR?
;9498  ;
;9499  ;-------------------------------------------------------------------------------
;9500  ;=0*
;9501  ;
;9502  ;CRC.ERROR6A:
;9503  ;
;9504  ;        NAD/XFER.DONE6            ;CRC ERROR. TERMINATE AND SET CRDY.
;9505  ;
;9506  ;-------------------------------------------------------------------------------
;9507  ;=1*
;9508  ;
;9509  ;NOT.CRC.ERR6A:
;9510  ;
;9511  ;        CLR.FIFO.ADR,             ;CLEAR THE FIFO ADDRESS COUNTER
;9512  ;        UCMD/SET.XFER.RQST,
;9513  ;        NAD/WAIT.XFER.GRANT6
;9514  ;
;9515  ;-------------------------------------------------------------------------------
;9516  ;=1*
;9517  ;
;9518  ;WAIT.XFER.GRANT6:
;9519  ;
;9520  ;        BEN1/XFER.REQ,            ;CPU TURNS ON CRDY IF LAST SECTOR
;9521  ;        NAD/WAIT.XFER.GRANT6      ;    THEN SENDS GRANT
;9522  ;
;9523  ;-------------------------------------------------------------------------------
;9524  ;=0*
;9525  ;
;9526  ;TEST.XFER.DONE6:
;9527  ;
;9528  ;        UCMD/CHG.FIFO,            ;SWITCH FIFOS, IN CASE THERE'S MORE DATA
;9529  ;        BEN0/CRDY.L,
;9530  ;        NAD/XFER.DONE6
;9531  ;
;9532  ;-------------------------------------------------------------------------------
;9533  ;=0
;9534  ;
;9535  ;XFER.DONE6:
;9536  ;
;9537  ;        UCMD/SET.CRDY,
;9538  ;        INCR.DAR/ON,
;9539  ;        FUNC/SET.INT,             ;CPU SIGNALS END OF TRANSFER.
;9540  ;        NAD/SET.INT.AND.IDLE
;9541  ;
;9542  ;-------------------------------------------------------------------------------
;9543  ;=111111110
;9544  ;
;9545  ;SET.INT.AND.IDLE:
;9546  ;
;9547  ;        FUNC/SET.INT,             ; The purpose of this state is to give CRDY one
```

```
;9548   ;       NAD/IDLE                        ; state to reset before asserting interrupt.
;9549   ;                                       ; CRDY is now used in the INT equation.
;9550   ;
;9551   ;-----------------------------------------------------------------------------
;9552   ;=1
;9553   ;
;9554   ;RTN.TO.IDLE:
;9555   ;IDLE:
;9556   ;
;9557   ;       BENO/CRDY.L,                    ;LOOK FOR GO (CRDY L)
;9558   ;       NAD/SETUP.DRV.CHK
;9559   ;-----------------------------------------------------------------------------
;9560   ;=0
;9561   ;
;9562   ;SETUP.DRV.CHK:
;9563   ;
;9564   ;       LOAD.LOOP.CNTR.[OFB],           ;COUNTER WILL BE USED
;9565   ;       NAD/SEL.NXT.DRV                 ;TO SEQUENCE THRU DRIVES.
;9566   ;
;9567   ;-----------------------------------------------------------------------------
;9568   ;
;9569   ;SEL.NXT.DRV:
;9570   ;
;9571   ;       SEL.DRV/UPROG,
;9572   ;       BENO/CNTR.OVFLW,
;9573   ;       NAD/CHK.DRV.STATUS
;9574   ;
;9575   ;-----------------------------------------------------------------------------
;9576   ;=0
;9577   ;
;9578   ;CHK.DRV.STATUS:
;9579   ;       ; Give drive status time to settle down.
;9580   ;
;9581   ;       SEL.DRV/UPROG,                  ;SELECT DRIVE
;9582   ;       NAD/DRV.STATUS.DISPATCH
;9583   ;
;9584   ;-----------------------------------------------------------------------------
;9585   ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
;9586   ;
;9587   ;DRV.STATUS.DISPATCH:
;9588   ;       SEL.DRV/UPROG,                  ;SELECT DRIVE
;9589   ;       BEN1/DRIVE.ONLINE,              ;AND BRANCH ON STATUS.
;9590   ;       BENO/DRIVE.RDY,
;9591   ;       NAD/-ERR.-OL.-RDY
;9592   ;
;9593   ;
;9594   ;-----------------------------------------------------------------------------
;9595   ;=011
;9596   ;
;9597   ;-ERR.OL.RDY:
;9598   ;
;9599   ;       SEL.DRV/UPROG,                  ;NOP
;9600   ;       INCR.LPCNTR/ON,                 ;ADD 1 TO DRIVE SELECT.
;9601   ;       NAD/SEL.NXT.DRV
;9602   ;
```

```
                  ;9603  ;--------------------------------------------------------------------------
                  ;9604  ;
                  ;9605  ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                  ;9606  ;
                  ;9607  ;               UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                  ;9608  ;
                  ;9609  ;--------------------------------------------------------------------------
                  ;9610  ;
                  ;9611  ;=1
                  ;9612  ;
                  ;9613  ;RTN.TO.IDLE:
                  ;9614  ;IDLE:
                  ;9615  ;
                  ;9616  ;       BENO/CRDY.L,              ;LOOK FOR GO (CRDY L)
                  ;9617  ;       NAD/SETUP.DRV.CHK
                  ;9618  ;
                  ;9619  ;--------------------------------------------------------------------------
                  ;9620
                  ;9621  ;Set up parameters for time-out error report :
                  ;9622
U 08E8, 36,4095   ;9623          MOV LS[#1] TO WR[1]
U 08E9, 3E,8295   ;9624          MOV WR[1] TO LS[ERROR.NUMBER]              ;set up error number = 1 in LS
U 08EA, E5,8A15   ;9625          CLR LS[ERROR.MASK]                         ;no error mask used here
U 08EB, 36,4A95   ;9626          MOV LS[IDC] TO WR[1]                       ;store module code in LS to
U 08EC, BE,8C95   ;9627          MOV WR[1] TO LS[MODULE.NUM]                ; modules
U 08ED, 36,6E95   ;9628          MOV LS[NA.EXP.REC] TO WR[1]                ;put "N/A" for printout data
U 08EE, BE,8095   ;9629          MOV WR[1] TO LS[CONTROL.STATUS]
                  ;9630
U 08EF, 2F,8015   ;9631          CLR WR[0]                                  ;Initialize time-out counter
U 08F0, BF,8C15   ;9632          MOV WR[0] TO LS[TIMEOUT.CNTR]
                  ;9633
                  ;9634  WAIT.IDC.T6.3:
U 08F1, 37,8C15   ;9635          MOV LS[TIMEOUT.CNTR] TO WR[0]              ;load the current time count
U 08F2, 37,9095   ;9636          MOV LS[WAIT.TIME1] TO WR[1]                ;time limit on read data
                  ;9637          CMP WR[0] WITH WR[1],                      ;timeout yet ???
U 08F3, 26,40B5   ;9638           DT(LONG)&SET.ALU.CC
U 08F4, 88,8FA9   ;9639          JMP.IF[EQL] TO [T6.ERR.1]                  ;yes, timeout on read data
U 08F5, 20,4015   ;9640          INC WR[0]                                  ;inc. time count
U 08F6, BF,8C15   ;9641          MOV WR[0] TO LS[TIMEOUT.CNTR]
U 08F7, DB,001A   ;9642          SKIP.IF[NO.FAST.INTERRUPT]                 ;wait for xfer req from IDC
U 08F8, 08,8FF4   ;9643          JMP [IDC.DONE.T6.2]                        ;yes, XFER REQ is set now
U 08F9, 88,8F14   ;9644          JMP [WAIT.IDC.T6.3]                        ;no, wait some more
                  ;9645
                  ;9646  ;Force an error in the CHECK.RESULT routine for this timeout :
                  ;9647
                  ;9648  T6.ERR.1:
U 08FA, 2F,8015   ;9649          CLR WR[0]                                  ;set unequal values
U 08FB, 36,4095   ;9650          MOV LS[#1] TO WR[1]                        ; between WR0 and WR1
U 08FC, 08,693C   ;9651          JSR [CHECK.RESULT]                         ;force an error
U 08FD, 88,8E34   ;9652          JMP [T.6.3]                                ;err & if LOE is set
U 08FE, 08,8FF4   ;9653          JMP [IDC.DONE.T6.2]
                  ;9654
                  ;9655  IDC.DONE.T6.2:
U 08FF, 90,FA15   ;9656          MISC2 [TRANSFER.GRANT]                     ;send grant to clr the xfer req
                  ;9657
                  ;9658  ;Now the home block is in FIFO A, but the 12-byte structure name is located
                  ;9659  ; at logical bytes " 216 to 227 (dec) " or " D8 to E3 (hex) "
                  ;9660
U 0900, 2F,8015   ;9661          CLR WR[0]                                 ;init. byte counter
U 0901, 17,D815   ;9662          PORT.CONTROL [SEL.FIFO.A]                 ;reading FIFO A
U 0902, 17,C015   ;9663          PORT.CONTROL [CLEAR.FIFO.CNTR]            ;ensure FIFO A adr.=0
U 0903, 90,9015   ;9664          MISC [SET.PORT.I.O]                       ;select idc
                  ;9665
U 0904, B7,EB15   ;9666          MOV LS[#D8] TO WR[2]                      ;FIFO adr. where label
                  ;9667                                                    ; begins
                  ;9668
                  ;9669  INC.FIFO.ADR.T6:                                  ;loop for step-up FIFO adr.
U 0905, 20,4015   ;9670          INC WR[0]                                 ;step up byte counter
U 0906, 17,8815   ;9671          PORT.READ PORT.ADRS[DATA.BYTE]            ; to get to next adr.
                  ;9672          CMP WR[0] WITH WR[2],                     ;reaching the boundary
U 0907, 26,4135   ;9673           DT(LONG)&SET.ALU.CC                      ; of the label yet ?
U 0908, 88,9051   ;9674          JMP.IF[NEQ] TO [INC.FIFO.ADR.T6]          ;no, get to the next adr.
U 0909, 88,90A4   ;9675          JMP [GET.LABEL.T6.1]
                  ;9676
                  ;9677
                  ;9678  ;Get the whole label and put them in 3 long words from LS(8E - 90)
                  ;9679
                  ;9680  GET.LABEL.T6.1:
U 090A, 08,716C   ;9681          JSR [LONG.WORD.LABEL]                     ;get 1st long word
U 090B, 36,0295   ;9682          MOV LS[SAV.WR0] TO WR[1]
U 090C, 3F,1C95   ;9683          MOV WR[1] TO LS[LABEL.READ1]
                  ;9684
                  ;9685  GET.LABEL.T6.2:
U 090D, 08,716C   ;9686          JSR [LONG.WORD.LABEL]                     ;get 2nd long word
U 090E, 36,0295   ;9687          MOV LS[SAV.WR0] TO WR[1]
U 090F, BF,1E95   ;9688          MOV WR[1] TO LS[LABEL.READ2]
                  ;9689
                  ;9690  GET.LABEL.T6.3:
U 0910, 08,716C   ;9691          JSR [LONG.WORD.LABEL]                     ;get 3rd long word
U 0911, 36,0295   ;9692          MOV LS[SAV.WR0] TO WR[1]
U 0912, 3F,2095   ;9693          MOV WR[1] TO LS[LABEL.READ3]
                  ;9694
                  ;9695
                  ;9696  ;Comparing the label "SCRATCH     " with FIFO A on a long word basis :
                  ;9697
                  ;9698  CMP.4CHAR.T6.1:                                   ;compare the 1st 4_char.
U 0913, 37,1C15   ;9699          MOV LS[LABEL.READ1] TO WR[0]              ; from the ones received
U 0914, 37,2915   ;9700          MOV LS[#53435241] TO WR[2]                ; with the expected "SCRA"
                  ;9701          CMP WR[0] WITH WR[2],
U 0915, 26,4135   ;9702           DT(LONG)&SET.ALU.CC
U 0916, 08,91F1   ;9703          JMP.IF[NEQ] TO [DIFF.LABEL]               ;wrong label detected
                  ;9704
                  ;9705  CMP.4CHAR.T6.2:                                   ;compare the 2nd 4_char.
U 0917, B7,1E15   ;9706          MOV LS[LABEL.READ2] TO WR[0]              ; from the ones received
U 0918, B6,2D95   ;9707          MOV LS[#FFFFFF00] TO WR[3]                ;the last ASCII is not cared
U 0919, AF,C615   ;9708          AND WR[3] TO WR[0]                        ; so it's masked out
U 091A, B7,2B15   ;9709          MOV LS[#54434820] TO WR[2]                ;mask out the last 2 ASCII
U 091B, 2F,C715   ;9710          AND WR[3] TO WR[2]                        ; of the expected "TCH "
                  ;9711          CMP WR[0] WITH WR[2],
U 091C, 26,4135   ;9712           DT(LONG)&SET.ALU.CC
```

```
U 091D, 08,91F1   ;9713           JMP.IF[NEQ] TO [DIFF.LABEL]            ;wrong label detected
U 091E, 08,9324   ;9714           JMP [RIGHT.LABEL]                      ;label compared
                  ;9715
                  ;9716   DIFF.LABEL:
U 091F, E5,8A15   ;9717           CLR LS[ERROR.MASK]                     ;no error mask for print out
U 0920, 36,4215   ;9718           MOV LS[#2] TO WR[0]
U 0921, BE,8215   ;9719           MOV WR[0] TO LS[ERROR.NUMBER]          ;update the error number = 2
U 0922, B6,4A15   ;9720           MOV LS[IDC] TO WR[0]                   ;identify the IDC & RL02 as
U 0923, C7,4C15   ;9721           BIS LS[RL02] TO WR[0]                  ; modules at fault and put that
U 0924, 3E,8C15   ;9722           MOV WR[0] TO LS[MODULE.NUM]            ;  in module number
                  ;9723
U 0925, 36,9015   ;9724           MOV LS[ERROR.CONTROL] TO WR[0]         ;look at the error control word
                  ;9725           BIT LS[APT.PRESENT] WITH WR[0],        ;see if it is running APT
U 0926, 59,4A35   ;9726            DT(LONG)&SET.ALU.CC
U 0927, 08,92D9   ;9727           JMP.IF[BITS.CLR] TO [ERR.ASCII.T6]     ;no, print the data in ASCII
                  ;9728
                  ;9729   ;Force the error for APT :
                  ;9730
U 0928, 2F,8015   ;9731           CLR WR[0]                              ;assign different
U 0929, 36,4095   ;9732           MOV LS[#1] TO WR[1]                    ; values for the WR's
U 092A, 08,693C   ;9733           JSR [CHECK.RESULT]                     ;go issue the error report
U 092B, 88,8E34   ;9734           JMP [T.6.3]                            ;error if LOE is set
U 092C, 08,9324   ;9735           JMP [END.T6]                           ;end of test
                  ;9736
                  ;9737   ERR.ASCII.T6:
U 092D, B6,4A15   ;9738           MOV LS[WARNING.IDC] TO WR[0]           ;set up the warning bit (5)
U 092E, 3E,8015   ;9739           MOV WR[0] TO LS[CONTROL.STATUS]        ; in the CS word
                  ;9740
U 092F, 10,E015   ;9741           MISC [SET.CP.ATTN]                     ;report the warning
                  ;9742
                  ;9743   WAIT.T6.1:
U 0930, 88,9304   ;9744           JMP [WAIT.T6.1]                        ;wait for console to respond
                  ;9745                                                  ; and halt
U 0931, 88,8E34   ;9746           JMP [T.6.3]                            ;error if LOE is set
                  ;9747
                  ;9748   RIGHT.LABEL:
                  ;9749   END.T6:                                        ;end of test
```

```
;9750  .PAGE  "Test 7  RL WRITE DATA FUNCTION TEST (not run in CRD mode)"
;9751  ;++
;9752  ;
;9753  ; Test description:
;9754  ;
;9755  ;        This test will seek to the innermost cylinder and write a sector of
;9756  ;        a data pattern to the disk by selecting the RL02 drive 1 and write
;9757  ;        data function <bits 3 and 1> of the CSR. The test will wait for a
;9758  ;        PORT XFER REQ signal from the IDC microcode to check the initiation
;9759  ;        of the write function. Then READ DATA function as described before
;9760  ;        is used to verify the data just written. This test will be repeated
;9761  ;        at the same disk address but with a different data pattern.
;9762  ;
;9763  ; Assumption:
;9764  ;
;9765  ;        Both CPU micros and IDC part I has been run successfully and disk
;9766  ;        drive 1 is up and spinning and ready to accept commands. And the
;9767  ;        seek, read header and read data functions are working properly.
;9768  ;
;9769  ;        This test assumes sector 0, track 0 & cylinder 511 has no bad block.
;9770  ;
;9771  ; Test Steps:
;9772  ;
;9773  ;         1. Seek to the innermost cylinder as outlined in "Read Header & Seek Tests"
;9774  ;         2. Obtain a data pattern from LS
;9775  ;         3. Port select FIFO A
;9776  ;         4. Clear the FIFO A address counter
;9777  ;         5. Write FIFO A with the data pattern
;9778  ;         6. Write the DAR by selecting sec. 2, tr. 0 & cyl. 511
;9779  ;         7. Write CSR with Drive 1 select & Write Data function bits
;9780  ;           <8, 3 & 1>
;9781  ;         8. Set up a timeout loop to wait for XFER REQ to set
;9782  ;         9. If timeout on write data function
;9783  ;        10.     Then setup error parameters and report ERROR 1
;9784  ;        11.     Else send TRANSFER GRANT to clear XFER REQ
;9785  ;        12. Port select FIFO A
;9786  ;        13. Clear the FIFO A address counter
;9787  ;        14. Write FIFO A with all zeros
;9788  ;        15. Write the DAR by selecting sec. 0, tr. 0 & cyl. 511
;9789  ;        16. Write CSR with the Drive 1 select & Read Data function
;9790  ;           bits <8, 3 & 2>
;9791  ;        17. Wait for XFER REQ to set
;9792  ;        18. Send TRANSFER GRANT to clear XFER REQ
;9793  ;        19. Get the data pattern again from the LS location
;9794  ;        20. Port select FIFO A
;9795  ;        21. Clear the FIFO A address counter
;9796  ;        22. Read FIFO A
;9797  ;        23. Compare the data received with the one expected
;9798  ;        24. If non-compare, setup error parameters and report ERROR 2
;9799  ;        25. Else repeat with a different data pattern once more
;9800  ;            (from step 1 to 24)
;9801  ;        26. Test ends
;9802  ;
;9803  ; Errors:
;9804  ;
```

```
                 ;9805  ;  ERROR 1 - Timeout error waiting for XFER REQ after a write data command
                 ;9806  ;
                 ;9807  ;         Printout:
                 ;9808  ;              EXPECTED        RECEIVED         MODULE
                 ;9809  ;              N/A             N/A              IDC
                 ;9810  ;
                 ;9811  ;  ERROR 2 - Data pattern non-compare
                 ;9812  ;
                 ;9813  ;         Printout:
                 ;9814  ;              EXPECTED        RECEIVED         MODULE
                 ;9815  ;              data pattern  data pattern     IDC RL02
                 ;9816  ;
                 ;9817  ; Debug:
                 ;9818  ;
                 ;9819  ;  ERROR 1 & 2: Check RL DRIVE READY, RL SYSTEM CLOCK, DS CLOCK, RL SECTOR
                 ;9820  ;                PLS, RL SEC PLS, RL READ DATA & RL WRITE DATA signals which
                 ;9821  ;                might have triggered timeout or data non-compare if at fault.
                 ;9822  ;
                 ;9823  ;--
                 ;9824
                 ;9825  T.7:
U 0932, B6,5E15  ;9826          MOV LS[BEGIN.TEST] TO WR[0]                 ;set bit 15 in WR[0] for
                 ;9827                                                      ; control and status word
U 0933, 3E,8015  ;9828          MOV WR[0] TO LS[CONTROL.STATUS]             ;set bit 15 in control and
                 ;9829                                                      ; status word - bit 15
                 ;9830                                                      ; indicates start of test to
                 ;9831                                                      ; console
U 0934, 10,E015  ;9832          MISC [SET.CP.ATTN]                          ;issue CPU ATTN to console
                 ;9833
                 ;9834  WAIT.T7.0:
U 0935, 88,9354  ;9835          JMP [WAIT.T7.0]                             ;loop here waiting for
                 ;9836                                                      ; console to respond
                 ;9837
U 0936, 65,A015  ;9838          CLR LS[PREVIOUS.ERROR]                      ;clear previous error number
                 ;9839                                                      ; in LS
U 0937, 17,CC15  ;9840          PORT.CONTROL [CLEAR.IDC]                    ;init. IDC
U 0938, 17,CC15  ;9841          PORT.CONTROL [CLEAR.IDC]
U 0939, 17,CC15  ;9842          PORT.CONTROL [CLEAR.IDC]
                 ;9843
                 ;9844  ;Position the heads to the innermost cylinder as described in test 5 before
                 ;9845  ; writing data to the disk :
                 ;9846
                 ;9847  T.7.RHEAD:
U 093A, B7,0615  ;9848          MOV LS[READ.HEADER] TO WR[0]                ;get the read header command
U 093B, 47,5015  ;9849          BIS LS[SETCSR.DS1] TO WR[0]                 ;select drive 1
U 093C, 17,A015  ;9850          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]        ;start to read the header
                 ;9851
                 ;9852          ;** refer to test 5 for IDC function codes **
                 ;9853
                 ;9854  WAIT.T.7.RHEAD:
U 093D, 88,700C  ;9855          JSR [WAIT.CSR.CRDY]                         ;check if CRDY set ?
U 093E, 08,9404  ;9856          JMP [T.7.SEEK]                              ;yes, go & do a seek
U 093F, 08,93D4  ;9857          JMP [WAIT.T.7.RHEAD]                        ;wait some more
                 ;9858
                 ;9859  ;Seek to the innermost cylinder :
                   ;9860
                   ;9861  T.7.SEEK:
U 0940, 88,72EC    ;9862          JSR [SHIFT.HDR]                         ;shift header to DAR format
U 0941, B7,EC95    ;9863          MOV LS[#0000FF80] TO WR[1]              ;max. cylinder for seek
U 0942, BF,8495    ;9864          MOV WR[1] TO LS[SEEK.CYL]               ;fetch cylinder to be sought
U 0943, 08,745C    ;9865          JSR [SEEK.DAR]                          ;setup the DAR word for seek
U 0944, B6,0215    ;9866          MOV LS[SAV.WR0] TO WR[0]                ;the right DAR for seek
U 0945, 97,A415    ;9867          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ;write to the DAR with this format
U 0946, B7,0495    ;9868          MOV LS[SEEK] TO WR[1]                   ;get the seek format
U 0947, C7,5095    ;9869          BIS LS[SETCSR.DS1] TO WR[1]             ;select drive 1
U 0948, 97,A095    ;9870          PORT.WRITE PORT.ADRS[CSR] WITH WR[1]    ;seek to the innermost cylinder
                   ;9871
                   ;9872          ;** refer to test 5 for IDC function codes **
                   ;9873
                   ;9874  WAIT.T.7.SEEK:
U 0949, 88,700C    ;9875          JSR [WAIT.CSR.CRDY]                     ;check if CRDY is set ?
U 094A, 5B,001E    ;9876          SKIP.IF[SKIP]                           ;start filling FIFO A
U 094B, 08,9494    ;9877          JMP [WAIT.T.7.SEEK]                     ;wait some more
                   ;9878
                   ;9879  CHECK.DRDY.T7.1:
U 094C, 08,708C    ;9880          JSR [WAIT.CSR.DRDY]                     ;wait for DRDY to set
                   ;9881
                   ;9882  ;Fill every address of FIFO A with a fixed data byte :
                   ;9883
U 094D, 36,9895    ;9884          MOV LS[#AAAAAAAA] TO WR[1]              ;load initial data pattern
U 094E, BE,0495    ;9885          MOV WR[1] TO LS[SAV.WR1]                ;save for later
U 094F, 65,0615    ;9886          CLR LS[SAV.WR2]                         ;init. test loop counter
                   ;9887
                   ;9888  T.7.2:
U 0950, 2F,8515    ;9889          CLR WR[2]                               ;FIFO A address counter
U 0951, 17,D815    ;9890          PORT.CONTROL [SEL.FIFO.A]               ;port select FIFO A
U 0952, 17,C015    ;9891          PORT.CONTROL [CLEAR.FIFO.CNTR]          ;init. FIFO counter
U 0953, 36,0495    ;9892          MOV LS[SAV.WR1] TO WR[1]                ;only the lowest data byte is
                   ;9893                                                  ; needed for this operation
                   ;9894  FILL.FIFO.T7.1:
U 0954, 17,A895    ;9895          PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[1] ;start writing one byte
U 0955, 20,4515    ;9896          INC WR[2]                               ;inc. the address counter
                   ;9897          BIT WR[2] WITH LS[BIT9],                ;see if it gets to 200 hex yet
U 0956, D9,5335    ;9898           DT(LONG)&SET.ALU.CC
U 0957, 88,9549    ;9899          JMP.IF[BITS.CLR] TO [FILL.FIFO.T7.1]    ;not yet, do the next byte
                   ;9900
                   ;9901  ;Now that FIFO A is full, start write data function to RL02 :
                   ;9902
                   ;9903  T.7.3:
U 0958, 37,EC15    ;9904          MOV LS[#0000FF80] TO WR[0]              ;load DAR with disk address of
U 0959, 97,A415    ;9905          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ; sec. 0, track 0 & cyl. 511
                   ;9906
U 095A, 37,0815    ;9907          MOV LS[WRITE.DATA] TO WR[0]             ;select write data function
U 095B, 47,5015    ;9908          BIS LS[SETCSR.DS1] TO WR[0]             ;select drive 1
U 095C, 17,A015    ;9909          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;write to the disk
                   ;9910
                   ;9911  ;*** RL02 Write Data IDC function codes : ***
                   ;9912
                   ;9913  ;--------------------------------------------------------------------------
                   ;9914  ;=1
```

```
;9915   ;
;9916   ;TEST.OR.RUN:
;9917   ;
;9918   ;        BEN2/MAINT,                  ;IS THAT YOU, VICKIE?
;9919   ;        NAD/DRIVE.TYPE
;9920   ;
;9921   ;-----------------------------------------------------------------------
;9922   ;=0**
;9923   ;
;9924   ;DRIVE.TYPE:
;9925   ;
;9926   ;        BENO/R80,                    ;WHICH TYPE OF DRIVE WAS SELECTED?
;9927   ;        NAD/RL02.IFORK
;9928   ;
;9929   ;-----------------------------------------------------------------------
;9930   ;=0
;9931   ;
;9932   ;RL02.IFORK:
;9933   ;
;9934   ;        UCMD/SET.FIFOA,
;9935   ;        FUNC/CLR.ECC.ERROR,          ; Enable clear of ECC/CRC errors for new function
;9936   ;        BENO/FO,                     ;8 WAY INSTRUCTION FORK.
;9937   ;        BEN1/F1,                     ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
;9938   ;        BEN2/F2,
;9939   ;        NAD/NOP.CMD
;9940   ;
;9941   ;-----------------------------------------------------------------------
;9942   ;=101
;9943   ;
;9944   ;WRT.DAT.CMD:
;9945   ;
;9946   ;        NAD/WAIT.DRDY1               ;START OF THE RL02 WRITE DATA COMMAND.
;9947   ;
;9948   ;-----------------------------------------------------------------------
;9949   ;=0
;9950   ;
;9951   ;WAIT.DRDY1:
;9952   ;
;9953   ;        BENO/DRIVE.RDY,              ;WAIT FOR DRIVE READY
;9954   ;        NAD/WAIT.DRDY1
;9955   ;
;9956   ;-----------------------------------------------------------------------
;9957   ;=1
;9958   ;
;9959   ;CHGCLK1:
;9960   ;
;9961   ;        ENB.SYS.CLK                  ;USE RL02 WRITE CLOCK
;9962   ;
;9963   ;-----------------------------------------------------------------------
;9964   ;=0*
;9965   ;
;9966   ;WAIT.SEC.PLS1:
;9967   ;
;9968   ;        BEN1/SECTOR.PLS,             ;WAIT FOR LEADING EDGE OF SECTOR PULSE
;9969   ;        NAD/WAIT.SEC.PLS1
```

```
;
;-------------------------------------------------------------------------
;=1*
;
;SEC.PLS.FND1:
;
;       NAD/WAIT.END.SEC.PLS1   ;SECTOR PULSE HAS BEEN FOUND
;
;-------------------------------------------------------------------------
;=1*
;
;WAIT.END.SEC.PLS1:
;
;       BEN1/SECTOR.PLS,        ;WAIT FOR END OF SECTOR PULSE
;       NAD/WAIT.END.SEC.PLS1
;
;-------------------------------------------------------------------------
;=0*
;
;SETUP.WAIT.PRE1:
;
;       LOAD.LOOP.CNTR.[0E0],   ;256-32=224=E0
;       NAD/WAIT.FOR.PREAMBLE1  ;SETUP DELAY BEFORE PREAMBLE.
;
;-------------------------------------------------------------------------
;=0
;
;WAIT.FOR.PREAMBLE1:
;
;       INCR.LPCNTR/ON,         ;WAIT FOR 8 MICROSECONDS
;       BEN0/CNTR.OVFLW,        ;TO REACH PREAMBLE.
;       NAD/WAIT.FOR.PREAMBLE1
;
;-------------------------------------------------------------------------
;=1
;
;SETUP.LOOP.LOCK1:
;
;       LOOP.LOCK/ON,           ;SETUP PHASE LOCK LOOP.
;       FUNC/CLR.ECC,
;       LOAD.LOOP.CNTR.[0E0],
;       NAD/WAIT.LOOP.LOCK1
;
;-------------------------------------------------------------------------
;=0
;
;WAIT.LOOP.LOCK1:
;
;       INCR.LPCNTR/ON,         ;ALLOW PHASE LOCK LOOP 4 BYTES
;       LOOP.LOCK/ON,           ;OF PREAMBLE TO LOCK UP.
;       UCON/CLR.MATCH,
;       BEN0/CNTR.OVFLW,
;       NAD/WAIT.LOOP.LOCK1
;
;-------------------------------------------------------------------------
```

```
;10025 ;=1
;10026 ;
;10027 ;SETUP.RD.HDR1:
;10028 ;
;10029 ;        LOOP.LOCK/ON,           ;SETUP INITIAL CONDITIONS
;10030 ;        ENB.CLR.BC/ON,          ;FOR READING HEADER.
;10031 ;        FUNC/CLR.DSR,
;10032 ;        LOAD.LOOP.CNTR.[0F1]    ;256-15=241=F1
;10033 ;
;10034 ;------------------------------------------------------------
;10035 ;
;10036 ;CHGCLK1A:
;10037 ;
;10038 ;        ENB.DSK.CLK,            ;THE MICROSEQUENCER WILL STALL
;10039 ;        LOOP.LOCK/ON,           ;UNTIL SYNC SEEN IS SET (SYNC BIT FOUND)
;10040 ;        ENB.CLR.BC/ON,          ;THEN WE USE CLOCK FROM DATA SEPARATOR.
;10041 ;        GEN.SYNC.[80]
;10042 ;
;10043 ;------------------------------------------------------------
;10044 ;=0
;10045 ;
;10046 ;READ.HDR1:
;10047 ;
;10048 ;        LOOP.LOCK/ON,           ;SHIFT IN 15 MORE HEADER BITS AND
;10049 ;        DATA.FIELD/ON,          ;COMPARE WITH SERIAL DAR.
;10050 ;        ENB.CLR.BC/ON,
;10051 ;        INCR.LPCNTR/ON,
;10052 ;        BENO/CNTR.OVFLW,
;10053 ;        NAD/READ.HDR1
;10054 ;
;10055 ;------------------------------------------------------------
;10056 ;=1
;10057 ;
;10058 ;TEST.HDR.MATCH1:
;10059 ;
;10060 ;        LOOP.LOCK/ON,           ;HEADER HAS BEEN READ.
;10061 ;        UCON/TEST.MATCH,        ;TURN ON TEST MATCH, OFF ENB CLR BC.
;10062 ;        DATA.FIELD/ON,
;10063 ;        NAD/BRANCH.HDR.MATCH1
;10064 ;
;10065 ;------------------------------------------------------------
;10066 ;
;10067 ;BRANCH.HDR.MATCH1:
;10068 ;
;10069 ;        LOOP.LOCK/ON,
;10070 ;        DATA.FIELD/ON,          ;RESULTS OF HDR COMP NOW IN MISMATCH.
;10071 ;        BEN2/MISMATCH.L,
;10072 ;        NAD/HDR.COMP1
;10073 ;
;10074 ;------------------------------------------------------------
;10075 ;=1**
;10076 ;
;10077 ;HDR.COMP1:
;10078 ;
;10079 ;        LOOP.LOCK/ON,           ;HEADER COMPARE!
```

```
;10080 ;        DATA.FIELD/ON,          ;CONTINUE TO READ THE HDR DENSITY WORD.
;10081 ;        LOAD.LOOP.CNTR.[OF2],   ;256-14=242=F2
;10082 ;        NAD/RD.DENSITY.WD1
;10083 ;
;10084 ;---------- --------------------------------------------------------
;10085 ;=0
;10086 ;
;10087 ;RD.DENSITY.WD1:
;10088 ;
;10089 ;        INCR.LPCNTR/ON,         ;CONT TO SHIFT IN THE HDR DENSITY WD.
;10090 ;        LOOP.LOCK/ON,
;10091 ;        DATA.FIELD/ON,
;10092 ;        BENO/CNTR.OVFLW,
;10093 ;        NAD/RD.DENSITY.WD1
;10094 ;
;10095 ;-------------------------------------------------------------------
;10096 ;=1
;10097 ;
;10098 ;CRC.PREP1:
;10099 ;
;10100 ;        LOAD.LOOP.CNTR.[OF1],   ;PREPARE TO READ THE CRC WORD.
;10101 ;        LOOP.LOCK/ON,           ;256-15=241=F1
;10102 ;        CHECK.FIELD/ON          ;TURN DFLD OFF AND CFLD ON.
;10103 ;
;10104 ;-------------------------------------------------------------------
;10105 ;=0
;10106 ;
;10107 ;RD.CRC.WD1:
;10108 ;
;10109 ;        INCR.LPCNTR/ON,         ;NOW WE SHIFT IN THE CRC WORD
;10110 ;        LOOP.LOCK/ON,
;10111 ;        CHECK.FIELD/ON,
;10112 ;        BENO/CNTR.OVFLW,
;10113 ;        NAD/RD.CRC.WD1
;10114 ;
;10115 ;-------------------------------------------------------------------
;10116 ;=1
;10117 ;
;10118 ;CHECK.CRC1:
;10119 ;
;10120 ;        LOOP.LOCK/ON,
;10121 ;        UCON/CHECK.ECC          ;DROP CFLD. CLOCK ECC FLOP
;10122 ;
;10123 ;-------------------------------------------------------------------
;10124 ;
;10125 ;TEST.FOR.ERR1:
;10126 ;
;10127 ;        LOOP.LOCK/ON,           ;WAS THERE A CRC ERROR?
;10128 ;        BEN1/ERROR.L
;10129 ;
;10130 ;-------------------------------------------------------------------
;10131 ;=1*
;10132 ;
;10133 ;NOT.CRC.ERR1:
;10134 ;
```

```
;10135 ;        LOOP.LOCK/ON,
;10136 ;        FUNC/CLR.ECC,           ;CLEAR DC007
;10137 ;        BEN2/F2,                ;BRANCH ON FUNCTION BITS INTO WRT CHK,
;10138 ;        BEN1/F1,                ;RD DATA OR WRT DATA FLOWS.
;10139 ;        NAD/BRANCH.WC.CMD1
;10140 ;
;10141 ;-----------------------------------------------------------------------
;10142 ;=10*
;10143 ;
;10144 ;BRANCH.WRT.CMD5:
;10145 ;
;10146 ;        LOOP.LOCK/ON,           ;CONTINUE VIA WRITE DATA FLOWS.
;10147 ;        BEN1/FIFO.OVFLW.L,      ;WAS FIFO FILLED BY CPU?
;10148 ;        BENO/DRIVE.RDY,         ;CHECK THAT DRDY DIDN'T GO AWAY.
;10149 ;        NAD/NOT.DRDY5
;10150 ;
;10151 ;-----------------------------------------------------------------------
;10152 ;=01
;10153 ;
;10154 ;CHGCLK5B:
;10155 ;
;10156 ;        LOOP.LOCK/ON,           ;EVERYTHING IS GO FOR WRITING DATA INTO
;10157 ;        ENB.SYS.CLK,            ;THIS SECTOR. RUN OFF SYSTEM CLOCK THRU
;10158 ;        LOAD.LOOP.CNTR.[0F8],   ;HEADER POSTAMBLE TO SPLICE LOCATION.
;10159 ;        NAD/HDR.POST5
;10160 ;
;10161 ;-----------------------------------------------------------------------
;10162 ;=0
;10163 ;
;10164 ;HDR.POST5:
;10165 ;
;10166 ;        INCR.LPCNTR/ON,         ;SHIFT OVER HEADER POSTAMBLE.
;10167 ;        BENO/CNTR.OVFLW,
;10168 ;        NAD/HDR.POST5
;10169 ;
;10170 ;-----------------------------------------------------------------------
;10171 ;=1
;10172 ;
;10173 ;SETUP.WRT.DAT5:
;10174 ;
;10175 ;        LOAD.LOOP.CNTR.[0D9],   ;PREPARE TO WRITE PREAMBLE.
;10176 ;        CLR.FIFO.ADR,
;10177 ;        FUNC/CLR.DSR,
;10178 ;        DISK.WRITE/ON
;10179 ;
;10180 ;-----------------------------------------------------------------------
;10181 ;=0
;10182 ;
;10183 ;WRT.DAT.PRE5:
;10184 ;
;10185 ;        DISK.WRITE/ON,          ;WRITE 5 BYTES (40) OF ZEROS.
;10186 ;        WRITE.GATE/ON,
;10187 ;        INCR.LPCNTR/ON,
;10188 ;        BENO/CNTR.OVFLW,
;10189 ;        NAD/WRT.DAT.PRE5
```

```
;10190 ;
;10191 ;
;10192 ;---------------------------------------------------------------
;10193 ;=1
;10194 ;
;10195 ;LOAD.SYNC.BYTE5:
;10196 ;
;10197 ;        DISK.WRITE/ON,          ;WRITE LAST 0 BIT.
;10198 ;        WRITE.GATE/ON,          ;LOAD SYNC BYTE 10000000.
;10199 ;        ENB.SYNC.[80],
;10200 ;        LOAD.DSR/ON
;10201 ;
;10202 ;---------------------------------------------------------------
;10203 ;
;10204 ;WRT.SYNC.BYTE5:
;10205 ;
;10206 ;        DISK.WRITE/ON,          ;WRITE SYNC BIT 0
;10207 ;        WRITE.GATE/ON,
;10208 ;        LOAD.LOOP.CNTR.[0FA]
;10209 ;
;10210 ;---------------------------------------------------------------
;10211 ;=0
;10212 ;
;10213 ;WRT.SYNC.BYTE5A:
;10214 ;
;10215 ;        DISK.WRITE/ON,          ;WRITE SYNC BITS 1 - 6
;10216 ;        WRITE.GATE/ON,
;10217 ;        INCR.LPCNTR/ON,
;10218 ;        BENO/CNTR.OVFLW,
;10219 ;        NAD/WRT.SYNC.BYTE5A
;10220 ;
;10221 ;---------------------------------------------------------------
;10222 ;=1
;10223 ;
;10224 ;LD.1ST.DAT.BYTE5:
;10225 ;
;10226 ;        DISK.WRITE/ON,          ;WRITE SYNC BIT 7.
;10227 ;        WRITE.GATE/ON,          ;LOAD FIRST DATA BYTE.
;10228 ;        ENB.FIFO/ON,
;10229 ;        LOAD.DSR/ON,
;10230 ;        INCR.FIFO.CNTR/ON,
;10231 ;        LOAD.LOOP.CNTR.[0F9],
;10232 ;        NAD/WRITE.DATA5
;10233 ;
;10234 ;---------------------------------------------------------------
;10235 ;=00
;10236 ;
;10237 ;WRT.LAST.DAT.BYTE5:
;10238 ;
;10239 ;        DISK.WRITE/ON,          ;WRITE LAST DATA BYTE (BITS 0 - 6)
;10240 ;        WRITE.GATE/ON,
;10241 ;        DATA.FIELD/ON,
;10242 ;        INCR.LPCNTR/ON,
;10243 ;        BENO/CNTR.OVFLW,
;10244 ;        NAD/WRT.LAST.DAT.BYTE5
```

```
;
;-----------------------------------------------------------------------------
;=01
;
;WRT.LAST.DAT.BIT5:
;
;        DISK.WRITE/ON,              ;WRITE LAST DATA BIT
;        WRITE.GATE/ON,
;        DATA.FIELD/ON,
;        LOAD.LOOP.CNTR.[0F1],
;        NAD/WRT.DAT.CRC5
;
;-----------------------------------------------------------------------------
;=10
;
;WRITE.DATA5:
;
;        DISK.WRITE/ON,              ;SHIFT OUT 7 DATA BITS (0 - 6)
;        WRITE.GATE/ON,
;        DATA.FIELD/ON,
;        INCR.LPCNTR/ON,
;        BEN0/CNTR.OVFLW,
;        NAD/WRITE.DATA5
;
;
;-----------------------------------------------------------------------------
;=11
;
;LD.NEXT.DAT.BYTE5:
;
;        DISK.WRITE/ON,              ;WRITE BIT 7 (LAST BIT IN BYTE)
;        WRITE.GATE/ON,
;        DATA.FIELD/ON,
;        ENB.FIFO/ON,
;        LOAD.DSR/ON,                ;LOAD NEXT DATA BYTE.
;        LOAD.LOOP.CNTR.[0F9],
;        INCR.FIFO.CNTR/ON,
;        BEN1/FIFO.MAX.L,
;        NAD/WRITE.DATA5
;
;-----------------------------------------------------------------------------
;=0
;
;WRT.DAT.CRC5:
;
;        DISK.WRITE/ON,              ;WRITE 16 BIT CRC
;        WRITE.GATE/ON,
;        CHECK.FIELD/ON,
;        INCR.LPCNTR/ON,
;        BEN0/CNTR.OVFLW,
;        NAD/WRT.DAT.CRC5
;
;-----------------------------------------------------------------------------
;=1
;
```

```
;WRT.DAT.CRC5A:
;
;        DISK.WRITE/ON,             ;WRITE LAST BIT OF CRC.
;        WRITE.GATE/ON,
;        CHECK.FIELD/ON,
;        LOAD.LOOP.CNTR.[0F0]
;
;-----------------------------------------------------------------------
;=0
;
;WRT.DAT.POST5:
;
;        DISK.WRITE/ON,             ;WRITE 16 ZEROS OF POSTAMBLE.
;        WRITE.GATE/ON,
;        INCR.LPCNTR/ON,
;        BEN0/CNTR.OVFLW,
;        NAD/WRT.DAT.POST5
;
;-----------------------------------------------------------------------
;=1
;
;CHGCLK5C:
;
;        ENB.CPU.CLK                ;SWITCH OVER FOR FAST INTR OR TERM.
;
;-----------------------------------------------------------------------
;
;SET.XFER.RQST5:
;
;        CLR.FIFO.ADR,              ;CLEAR FIFO COUNTER FOR CPU.
;        UCMD/SET.XFER.RQST,
;        NAD/WAIT.XFER.GRANT5
;
;-----------------------------------------------------------------------
;=0*
;
;TEST.XFER.DONE5:
;
;        UCMD/CHG.FIFO,             ;SWITCH FIFOS, IN CASE THERE'S MORE DATA
;        BEN0/CRDY.L,
;        NAD/XFER.DONE5
;
;
;-----------------------------------------------------------------------
;=1*
;
;WAIT.XFER.GRANT5:
;
;        BEN1/XFER.REQ,             ;CPU SIGNALS IF LAST SECTOR.
;        NAD/WAIT.XFER.GRANT5       ;THEN SENDS GRANT
;
;-----------------------------------------------------------------------
;=0
;
;XFER.DONE5:
```

```
;10355 ;
;10356 ;        UCMD/SET.CRDY,
;10357 ;        INCR.DAR/ON,
;10358 ;        FUNC/SET.INT,               ;CPU SIGNALLED THAT TRANSFER IS OVER.
;10359 ;        NAD/SET.INT.AND.IDLE
;10360 ;
;10361 ;------------------------------------------------------------------------
;10362 ;=111111110
;10363 ;
;10364 ;SET.INT.AND.IDLE:
;10365 ;
;10366 ;        FUNC/SET.INT,               ; The purpose of this state is to give CRDY one
;10367 ;        NAD/IDLE                    ; state to reset before asserting interrupt.
;10368 ;                                    ; CRDY is now used in the INT equation.
;10369 ;
;10370 ;------------------------------------------------------------------------
;10371 ;=1
;10372 ;
;10373 ;RTN.TO.IDLE:
;10374 ;IDLE:
;10375 ;
;10376 ;        BENO/CRDY.L,                ;LOOK FOR GO (CRDY L)
;10377 ;        NAD/SETUP.DRV.CHK
;10378 ;
;10379 ;------------------------------------------------------------------------
;10380 ;=0
;10381 ;
;10382 ;SETUP.DRV.CHK:
;10383 ;
;10384 ;        LOAD.LOOP.CNTR.[0FB],       ;COUNTER WILL BE USED
;10385 ;        NAD/SEL.NXT.DRV             ;TO SEQUENCE THRU DRIVES.
;10386 ;
;10387 ;------------------------------------------------------------------------
;10388 ;
;10389 ;SEL.NXT.DRV:
;10390 ;
;10391 ;        SEL.DRV/UPROG,
;10392 ;        BENO/CNTR.OVFLW,
;10393 ;        NAD/CHK.DRV.STATUS
;10394 ;
;10395 ;------------------------------------------------------------------------
;10396 ;=0
;10397 ;
;10398 ;CHK.DRV.STATUS:
;10399 ;        ; Give drive status time to settle down.
;10400 ;
;10401 ;        SEL.DRV/UPROG,              ;SELECT DRIVE
;10402 ;        NAD/DRV.STATUS.DISPATCH
;10403 ;
;10404 ;------------------------------------------------------------------------
;10405 ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
;10406 ;
;10407 ;DRV.STATUS.DISPATCH:
;10408 ;        SEL.DRV/UPROG,              ;SELECT DRIVE
;10409 ;        BEN1/DRIVE.ONLINE,          ;AND BRANCH ON STATUS.
```

```
                     ;10410 ;        BENO/DRIVE.RDY,
                     ;10411 ;        NAD/-ERR.-OL.-RDY
                     ;10412 ;
                     ;10413 ;
                     ;10414 ;------------------------------------------------------------------------
                     ;10415 ;=011
                     ;10416 ;
                     ;10417 ;-ERR.OL.RDY:
                     ;10418 ;
                     ;10419 ;        SEL.DRV/UPROG,          ;NOP
                     ;10420 ;        INCR.LPCNTR/ON,         ;ADD 1 TO DRIVE SELECT.
                     ;10421 ;        NAD/SEL.NXT.DRV
                     ;10422 ;
                     ;10423 ;------------------------------------------------------------------------
                     ;10424 ;
                     ;10425 ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                     ;10426 ;
                     ;10427 ;               UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                     ;10428 ;
                     ;10429 ;------------------------------------------------------------------------
                     ;10430 ;
                     ;10431 ;=1
                     ;10432 ;
                     ;10433 ;RTN.TO.IDLE:
                     ;10434 ;IDLE:
                     ;10435 ;
                     ;10436 ;        BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
                     ;10437 ;        NAD/SETUP.DRV.CHK
                     ;10438 ;
                     ;10439 ;------------------------------------------------------------------------
                     ;10440
                     ;10441 ;Set up parameters for time-out error report :
                     ;10442
U 095D, 36,4095      ;10443         MOV LS[#1] TO WR[1]
U 095E, 3E,8295      ;10444         MOV WR[1] TO LS[ERROR.NUMBER]          ;set up error number = 1 in LS
U 095F, E5,8A15      ;10445         CLR LS[ERROR.MASK]                     ;no error mask used here
U 0960, 36,4A95      ;10446         MOV LS[IDC] TO WR[1]                   ;store module code in LS to
U 0961, BE,8C95      ;10447         MOV WR[1] TO LS[MODULE.NUM]            ; modules
U 0962, 36,6E95      ;10448         MOV LS[NA.EXP.REC] TO WR[1]            ;put "N/A" for printout data
U 0963, BE,8095      ;10449         MOV WR[1] TO LS[CONTROL.STATUS]
                     ;10450
U 0964, 2F,8015      ;10451         CLR WR[0]                              ;Initialize time-out counter
U 0965, BF,8C15      ;10452         MOV WR[0] TO LS[TIMEOUT.CNTR]
                     ;10453
                     ;10454 ;Timeout loop to wait for XFER REQ after write data has been issued :
                     ;10455
                     ;10456 WAIT.IDC.T7.1:
U 0966, 37,8C15      ;10457         MOV LS[TIMEOUT.CNTR] TO WR[0]          ;load the current time count
U 0967, 37,9095      ;10458         MOV LS[WAIT.TIME1] TO WR[1]            ;time limit on write data
                     ;10459         CMP WR[0] WITH WR[1],                  ;timeout yet ???
U 0968, 26,40B5      ;10460          DT(LONG)&SET.ALU.CC
U 0969, 08,96F9      ;10461         JMP.IF[EQL] TO [T7.ERR.1]              ;yes, timeout on write data
U 096A, 20,4015      ;10462         INC WR[0]                              ;inc. time count
U 096B, BF,8C15      ;10463         MOV WR[0] TO LS[TIMEOUT.CNTR]
U 096C, DB,001A      ;10464         SKIP.IF[NO.FAST.INTERRUPT]             ;check if XFER REQ is set
```

```
U 096D, 88,9744   ;10465          JMP [IDC.DONE.T7.1]                    ;yes, XFER REQ is set now
U 096E, 88,9664   ;10466          JMP [WAIT.IDC.T7.1]                    ;no, wait some more
                  ;10467
                  ;10468 ;Force an error in the CHECK.RESULT routine for this timout :
                  ;10469
                  ;10470 T7.ERR.1:
U 096F, 2F,8015   ;10471          CLR WR[0]                              ;set unequal values
U 0970, 36,4095   ;10472          MOV LS[#1] TO WR[1]                    ; between WR0 and WR1
U 0971, 08,693C   ;10473          JSR [CHECK.RESULT]                     ;force an error
U 0972, 08,9584   ;10474          JMP [T.7.3]                            ;err and if LOE is set
U 0973, 88,9744   ;10475          JMP [IDC.DONE.T7.1]
                  ;10476
                  ;10477 IDC.DONE.T7.1:
U 0974, 90,FA15   ;10478          MISC2 [TRANSFER.GRANT]                 ;send grant to clr XFER REQ
                  ;10479
                  ;10480 ;Before doing a read data command, it is necessary to ensure that the selected
                  ;10481 ; FIFO (FIFO A in this case) is empty:
                  ;10482
                  ;10483 T.7.4:
U 0975, 2F,8295   ;10484          CLR WR[1]                              ;FIFO A address counter
U 0976, 17,D815   ;10485          PORT.CONTROL [SEL.FIFO.A]              ;port select FIFO A
U 0977, 17,C015   ;10486          PORT.CONTROL [CLEAR.FIFO.CNTR]         ;init. FIFO counter
U 0978, 36,9C15   ;10487          MOV LS[#0] TO WR[0]                    ; with all zeros
                  ;10488
                  ;10489 FILL.FIFO.T7.2:
U 0979, 97,A815   ;10490          PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;start writing one byte
U 097A, 20,4295   ;10491          INC WR[1]                              ;inc. the address counter
                  ;10492          BIT WR[1] WITH LS[BIT9],               ;see if it gets to 200 hex yet
U 097B, D9,52B5   ;10493           DT(LONG)&SET.ALU.CC
U 097C, 88,9799   ;10494          JMP.IF[BITS.CLR] TO [FILL.FIFO.T7.2]   ;not yet, do the next byte
                  ;10495
                  ;10496 ;Now that FIFO A is empty, start read data function to RL02 :
                  ;10497
U 097D, 37,EC15   ;10498          MOV LS[#0000FF80] TO WR[0]             ;load DAR with disk address of
U 097E, 97,A415   ;10499          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]   ; sec. 0, track 0 & cyl. 511
                  ;10500
U 097F, B7,0A15   ;10501          MOV LS[READ.DATA] TO WR[0]             ;select read data function
U 0980, 47,5015   ;10502          BIS LS[SETCSR.DS1] TO WR[0]            ;select drive 1
U 0981, 17,A015   ;10503          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]   ;write to the disk
                  ;10504
                  ;10505          ;*****************************************************************
                  ;10506          ;* REFER TO THE IDC READ DATA FUNCTION CODES AS LISTED IN TEST 6 *
                  ;10507          ;*****************************************************************
                  ;10508
                  ;10509 WAIT.IDC.T7.2:
U 0982, DB,001A   ;10510          SKIP.IF[NO.FAST.INTERRUPT]             ;wait for XFER REQ from IDC
U 0983, 08,9854   ;10511          JMP [IDC.DONE.T7.2]                    ;XFER REQ set
U 0984, 88,9824   ;10512          JMP [WAIT.IDC.T7.2]                    ;wait some more
                  ;10513
                  ;10514 ;Set up error parameters for ERROR 2 report :
                  ;10515
                  ;10516 IDC.DONE.T7.2:
U 0985, 90,FA15   ;10517          MISC2 [TRANSFER.GRANT]
U 0986, 36,4215   ;10518          MOV LS[#2] TO WR[0]                    ;error number = 2
U 0987, BE,8215   ;10519          MOV WR[0] TO LS[ERROR.NUMBER]
```

```
U 0988, B6,2C15   ;10520          MOV LS[#FFFFFF00] TO WR[0]              ;error mask for RL02
U 0989, 3E,8A15   ;10521          MOV WR[0] TO LS[ERROR.MASK]             ; expected status
U 098A, 36,4A95   ;10522          MOV LS[IDC] TO WR[1]                    ;store IDC and RL02 in LS
U 098B, 47,4C95   ;10523          BIS LS[RL02] TO WR[1]                   ; as module codes to indicate
U 098C, BE,8C95   ;10524          MOV WR[1] TO LS[MODULE.NUM]             ;  modules at fault
U 098D, E5,8015   ;10525          CLR LS[CONTROL.STATUS]                  ; clear [NA.EXP.REC]
                  ;10526
                  ;10527 ;Read the data from FIFO A and compare the data :
                  ;10528
                  ;10529 IDC.DONE.T7.3:
U 098E, 17,D815   ;10530          PORT.CONTROL [SEL.FIFO.A]               ;read FIFO A
U 098F, 17,C015   ;10531          PORT.CONTROL [CLEAR.FIFO.CNTR]          ;clear FIFO A address
U 0990, 90,9015   ;10532          MISC [SET.PORT.I.O]                     ;select IDC
U 0991, 17,8815   ;10533          PORT.READ PORT.ADRS[DATA.BYTE]          ;read data from the FIFO
U 0992, 3B,3215   ;10534          MOV PORT TO LS[PORT0]
U 0993, 37,3215   ;10535          MOV LS[PORT0] TO WR[0]                  ;load WR0 with the data
U 0994, 36,0495   ;10536          MOV LS[SAV.WR1] TO WR[1]                ;expected status
U 0995, 08,693C   ;10537          JSR [CHECK.RESULT]                      ;do the data compare ??
U 0996, 08,9754   ;10538          JMP [T.7.4]                             ;err and if LOE is set
U 0997, 08,9984   ;10539          JMP [T.7.5]                             ;test done !!
                  ;10540
                  ;10541 ;Go do the entire test with a different data pattern one more time if the test
                  ;10542 ; is done once :
                  ;10543
                  ;10544 T.7.5:
U 0998, B6,0715   ;10545          MOV LS[SAV.WR2] TO WR[2]                ;fetch the test loop counter
                  ;10546          BIT WR[2] WITH LS[BIT0],                ;is this the first test loop ??
U 0999, D9,4135   ;10547           DT(LONG)&SET.ALU.CC
U 099A, 5B,0009   ;10548          SKIP.IF[BITS.CLR]                       ;yes, first time
U 099B, 08,9A14   ;10549          JMP [END.T7]                            ;no, test ends
U 099C, 20,4515   ;10550          INC WR[2]                               ;inc. the test loop counter
U 099D, 3E,0715   ;10551          MOV WR[2] TO LS[SAV.WR2]                ;and flag it
U 099E, B6,9A95   ;10552          MOV LS[#55555555] TO WR[1]              ;load a different data pattern
U 099F, BE,0495   ;10553          MOV WR[1] TO LS[SAV.WR1]                ;store it up for later
U 09A0, 88,9504   ;10554          JMP [T.7.2]                             ;do the test once more !!
                  ;10555
                  ;10556 END.T7:                                          ;end of test 7
                  ;10557
```

```
.PAGE  "Test 8  RL WRITE CHECK DATA TEST (not run in CRD mode)"
;++
;
; Test description:
;
;          This test will verify the RL WRITE CHECK function by first writing
;          data to the disk similar to the operation procedure as in the "RL
;          WRITE DATA FUNCTION TEST" described above. And then the test will
;          load the data buffer, i.e. FIFO A, with the data just used in the
;          write data function, and assert the CSR <bit 1>. The test will wait
;          for a PORT XFER REQ signal from the IDC microcode as a sign of no
;          write check error.
;
; Assumption:
;
;          Both CPU micros and IDC part I has been run successfully and disk
;          drive 1 is up and spinning and ready to accept commands. And the
;          seek, read header, read data and write data functions are working
;          properly.
;
; Test steps:
;
;            1. Seek to the maximum cylinder as outlined in "Read Header & Seek Tests"
;            2. Obtain a data pattern from LS
;            3. Port select FIFO A
;            4. Clear the FIFO A address counter
;            5. Write FIFO A with the data pattern
;            6. Write the DAR by selecting sec. 2, tr. 0 & cyl. 511
;            7. Write CSR with Drive 1 select & Write Data function bits
;               <8, 3 & 1>
;            8. Wait for XFER REQ to set
;            9. Send TRANSFER GRANT to clear XFER REQ
;           10. Port select FIFO A
;           11. Clear the FIFO A address counter
;           12. Write FIFO A with the data pattern
;           13. Write the DAR by selecting sec. 2, tr. 0 & cyl. 511
;           14. Write CSR with Drive 1 select & Write Check function bits
;               <8 & 1>
;           15. Set up a timeout loop for XFER REQ set
;           16. If timeout on write check command
;           17.        Then setup error parameters, read the CSR status and put that
;                          under "OTHER DATA" and report ERROR 2
;           18.        Else
;           19. Test ends
;
; Error:
;
;   ERROR 1 - Time out waiting for XFER REQ signal after a Write Check
;             command has been issued to the IDC
;
;       Printout:
;             EXPECTED         RECEIVED          OTHER        MODULE
;             N/A              N/A               CSR          IDC
;                                                status
;
```

```
                   ;10613 ; Debug:
                   ;10614 ;
                   ;10615 ;   ERROR 1: Check DRIVE READY, RL SYSTEM CLOCK, SYNC SEEN, DS CLOCK,
                   ;10616 ;            RL READ DATA & WRT CHK LOAD for fault. Check PAL 16R4
                   ;10617 ;            (DSK RD SEQ REG) and its signals: DSR 0, COMPARE,
                   ;10618 ;            UWRT CHK, WRTCHK LOAD DSR.
                   ;10619 ;
                   ;10620 ;--
                   ;10621
                   ;10622 T.8:
U 09A1, B6,5E15    ;10623         MOV LS[BEGIN.TEST] TO WR[0]              ;set bit 15 in WR[0] for
                   ;10624                                                  ; control and status word
U 09A2, 3E,8015    ;10625         MOV WR[0] TO LS[CONTROL.STATUS]          ;set bit 15 in control and
                   ;10626                                                  ; status word - bit 15
                   ;10627                                                  ; indicates start of test to
                   ;10628                                                  ; console
U 09A3, 10,E015    ;10629         MISC [SET.CP.ATTN]                       ;issue CPU ATTN to console
                   ;10630
                   ;10631 WAIT.T8.0:
U 09A4, 08,9A44    ;10632         JMP [WAIT.T8.0]                          ;loop here waiting for
                   ;10633                                                  ; console to respond
                   ;10634
U 09A5, 65,A015    ;10635         CLR LS[PREVIOUS.ERROR]                   ;clear previous error number
                   ;10636                                                  ; in LS
U 09A6, 17,CC15    ;10637         PORT.CONTROL [CLEAR.IDC]                 ;init. IDC
U 09A7, 17,CC15    ;10638         PORT.CONTROL [CLEAR.IDC]
U 09A8, 17,CC15    ;10639         PORT.CONTROL [CLEAR.IDC]
                   ;10640
                   ;10641 ;Position the heads to the innermost cylinder as described in test 5 before
                   ;10642 ; writing data to the disk :
                   ;10643
                   ;10644 T.8.RHEAD:
U 09A9, B7,0615    ;10645         MOV LS[READ.HEADER] TO WR[0]             ;get the read header command
U 09AA, 47,5015    ;10646         BIS LS[SETCSR.DS1] TO WR[0]              ;select drive 1
U 09AB, 17,A015    ;10647         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]     ;start to read the header
                   ;10648
                   ;10649         ;** refer to test 5 for IDC function codes **
                   ;10650
                   ;10651 WAIT.T.8.RHEAD:
U 09AC, 88,700C    ;10652         JSR [WAIT.CSR.CRDY]                      ;check if CRDY set ?
U 09AD, 88,9AF4    ;10653         JMP [T.8.SEEK]                           ;yes, go & do a seek
U 09AE, 88,9AC4    ;10654         JMP [WAIT.T.8.RHEAD]                     ;wait some more
                   ;10655
                   ;10656 ;Seek to the innermost cylinder :
                   ;10657
                   ;10658 T.8.SEEK:
U 09AF, 88,72EC    ;10659         JSR [SHIFT.HDR]                          ;shift header to DAR format
U 09B0, B7,EC95    ;10660         MOV LS[#0000FF80] TO WR[1]               ;max. cylinder for seek
U 09B1, BF,8495    ;10661         MOV WR[1] TO LS[SEEK.CYL]                ;fetch cylinder to be sought
U 09B2, 08,745C    ;10662         JSR [SEEK.DAR]                           ;setup the DAR word for seek
U 09B3, B6,0215    ;10663         MOV LS[SAV.WR0] TO WR[0]                 ;the right DAR for seek
U 09B4, 97,A415    ;10664         PORT.WRITE PORT.ADRS[DAR] WITH WR[0]     ;write to the DAR with this format
U 09B5, B7,0495    ;10665         MOV LS[SEEK] TO WR[1]                    ;get the seek format
U 09B6, C7,5095    ;10666         BIS LS[SETCSR.DS1] TO WR[1]              ;select drive 1
U 09B7, 97,A095    ;10667         PORT.WRITE PORT.ADRS[CSR] WITH WR[1]     ;seek to the innermost cylinder
```

```
                     ;10668
                     ;10669          ;** refer to test 5 for IDC function codes **
                     ;10670
                     ;10671 WAIT.T.8.SEEK:
U 09B8, 88,700C      ;10672          JSR [WAIT.CSR.CRDY]                       ;check if CRDY is set ?
U 09B9, 5B,001E      ;10673          SKIP.IF[SKIP]                             ;start filling FIFO A
U 09BA, 88,9B84      ;10674          JMP [WAIT.T.8.SEEK]                       ;wait some more
                     ;10675
                     ;10676 ;Fill every address of FIFO A with a fixed data byte :
                     ;10677
                     ;10678 T.8.2:
U 09BB, 2F,8295      ;10679          CLR WR[1]                                 ;FIFO A address counter
U 09BC, 17,D815      ;10680          PORT.CONTROL [SEL.FIFO.A]                 ;port select FIFO A
U 09BD, 17,C015      ;10681          PORT.CONTROL [CLEAR.FIFO.CNTR]            ;init. FIFO counter
U 09BE, 36,9A15      ;10682          MOV LS[#55555555] TO WR[0]                ;load initial data pattern
                     ;10683
                     ;10684 FILL.FIFO.T8.1:
U 09BF, 97,A815      ;10685          PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;start writing one byte
U 09C0, 20,4295      ;10686          INC WR[1]                                 ;inc. the address counter
                     ;10687          BIT WR[1] WITH LS[BIT9],                  ;see if it gets to 200 hex yet
U 09C1, D9,52B5      ;10688           DT(LONG)&SET.ALU.CC
U 09C2, 88,9BF9      ;10689          JMP.IF[BITS.CLR] TO [FILL.FIFO.T8.1]      ;not yet, do the next byte
                     ;10690
                     ;10691 ;Now that FIFO A is full, start writing data function to RL02 :
                     ;10692
                     ;10693 T.8.3:
U 09C3, 37,EC15      ;10694          MOV LS[#0000FF80] TO WR[0]                ;load DAR with disk address of
U 09C4, 97,A415      ;10695          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]      ; sec. 0, track 0 & cyl. 511
                     ;10696
U 09C5, 37,0815      ;10697          MOV LS[WRITE.DATA] TO WR[0]               ;select write data function
U 09C6, 47,5015      ;10698          BIS LS[SETCSR.DS1] TO WR[0]               ;select drive 1
U 09C7, 17,A015      ;10699          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;write to the disk
                     ;10700
                     ;10701          ;*****************************************************************
                     ;10702          ;* REFER TO THE WRITE DATA IDC FUNCTION CODES AS LISTED IN TEST 7 *
                     ;10703          ;*****************************************************************
                     ;10704
                     ;10705 WAIT.IDC.T8.1:
U 09C8, DB,001A      ;10706          SKIP.IF[NO.FAST.INTERRUPT]                ;wait for XFER REQ from IDC
U 09C9, 08,9CB4      ;10707          JMP [IDC.DONE.T8.1]                       ;XFER REQ set
U 09CA, 08,9C84      ;10708          JMP [WAIT.IDC.T8.1]                       ;wait some more
                     ;10709
                     ;10710 ;Send TRANSFER GRANT to IDC and clear FIFO A address counter :
                     ;10711
                     ;10712 IDC.DONE.T8.1:
U 09CB, 90,FA15      ;10713          MISC2 [TRANSFER.GRANT]                    ;reset XFER REQ
                     ;10714
                     ;10715 ;Refill every address of FIFO A with the same data byte :
                     ;10716
                     ;10717 T.8.4:
U 09CC, 2F,8295      ;10718          CLR WR[1]                                 ;FIFO A address counter
U 09CD, 17,D815      ;10719          PORT.CONTROL [SEL.FIFO.A]                 ;port select FIFO A
U 09CE, 17,C015      ;10720          PORT.CONTROL [CLEAR.FIFO.CNTR]            ;init. FIFO counter
U 09CF, 36,9A15      ;10721          MOV LS[#55555555] TO WR[0]                ;load initial data pattern
                     ;10722
```

```
                    ;10723 FILL.FIFO.T8.2:
U 09D0, 97,A815     ;10724          PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;start writing one byte
U 09D1, 20,4295     ;10725          INC WR[1]                               ;inc. the address counter
                    ;10726          BIT WR[1] WITH LS[BIT9],                ;see if it gets to 200 hex yet
U 09D2, D9,52B5     ;10727           DT(LONG)&SET.ALU.CC
U 09D3, 88,9D09     ;10728          JMP.IF[BITS.CLR] TO [FILL.FIFO.T8.2]    ;not yet, do the next byte
                    ;10729
                    ;10730 ;Now that FIFO A is full, perform a write check command at the same disk address :
                    ;10731
U 09D4, 37,EC15     ;10732          MOV LS[#0000FF80] TO WR[0]              ;fetch max. cylinder address
U 09D5, 97,A415     ;10733          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ;write to the DAR
                    ;10734
U 09D6, B7,0015     ;10735          MOV LS[WRITE.CHECK] TO WR[0]            ;fetch the write check function
U 09D7, 47,5015     ;10736          BIS LS[SETCSR.DS1] TO WR[0]             ;select drive 1
U 09D8, 17,A015     ;10737          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;issue the write check command
                    ;10738
                    ;10739 ;*** RL02 WRITE CHECK IDC FUNCTION CODES : ***
                    ;10740
                    ;10741 ;-----------------------------------------------------------------------
                    ;10742 ;=1
                    ;10743 ;
                    ;10744 ;TEST.OR.RUN:
                    ;10745 ;
                    ;10746 ;        BEN2/MAINT,             ;IS THAT YOU, VICKIE?
                    ;10747 ;        NAD/DRIVE.TYPE
                    ;10748 ;
                    ;10749 ;-----------------------------------------------------------------------
                    ;10750 ;=0**
                    ;10751 ;
                    ;10752 ;DRIVE.TYPE:
                    ;10753 ;
                    ;10754 ;        BEN0/R80,               ;WHICH TYPE OF DRIVE WAS SELECTED?
                    ;10755 ;        NAD/RL02.IFORK
                    ;10756 ;
                    ;10757 ;-----------------------------------------------------------------------
                    ;10758 ;=0
                    ;10759 ;
                    ;10760 ;RL02.IFORK:
                    ;10761 ;
                    ;10762 ;        UCMD/SET.FIFOA,
                    ;10763 ;        FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
                    ;10764 ;        BEN0/F0,                ;8 WAY INSTRUCTION FORK.
                    ;10765 ;        BEN1/F1,                ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                    ;10766 ;        BEN2/F2,
                    ;10767 ;        NAD/NOP.CMD
                    ;10768 ;
                    ;10769 ;-----------------------------------------------------------------------
                    ;10770 ;=101
                    ;10771 ;
                    ;10772 ;WRT.DAT.CMD:
                    ;10773 ;
                    ;10774 ;        NAD/WAIT.DRDY1          ;START OF THE RL02 WRITE DATA COMMAND.
                    ;10775 ;
                    ;10776 ;-----------------------------------------------------------------------
                    ;10777 ;=0
```

```
;10778 ;
;10779 ;WAIT.DRDY1:
;10780 ;
;10781 ;        BENO/DRIVE.RDY,           ;WAIT FOR DRIVE READY
;10782 ;        NAD/WAIT.DRDY1
;10783 ;
;10784 ;------------------------------------------------------------------
;10785 ;=1
;10786 ;
;10787 ;CHGCLK1:
;10788 ;
;10789 ;        ENB.SYS.CLK               ;USE RL02 WRITE CLOCK
;10790 ;
;10791 ;------------------------------------------------------------------
;10792 ;=0*
;10793 ;
;10794 ;WAIT.SEC.PLS1:
;10795 ;
;10796 ;        BEN1/SECTOR.PLS,          ;WAIT FOR LEADING EDGE OF SECTOR PULSE
;10797 ;        NAD/WAIT.SEC.PLS1
;10798 ;
;10799 ;------------------------------------------------------------------
;10800 ;=1*
;10801 ;
;10802 ;SEC.PLS.FND1:
;10803 ;
;10804 ;        NAD/WAIT.END.SEC.PLS1    ;SECTOR PULSE HAS BEEN FOUND
;10805 ;
;10806 ;------------------------------------------------------------------
;10807 ;=1*
;10808 ;
;10809 ;WAIT.END.SEC.PLS1:
;10810 ;
;10811 ;        BEN1/SECTOR.PLS,          ;WAIT FOR END OF SECTOR PULSE
;10812 ;        NAD/WAIT.END.SEC.PLS1
;10813 ;
;10814 ;------------------------------------------------------------------
;10815 ;=0*
;10816 ;
;10817 ;SETUP.WAIT.PRE1:
;10818 ;
;10819 ;        LOAD.LOOP.CNTR.[0E0],     ;256-32=224=E0
;10820 ;        NAD/WAIT.FOR.PREAMBLE1   ;SETUP DELAY BEFORE PREAMBLE.
;10821 ;
;10822 ;------------------------------------------------------------------
;10823 ;=0
;10824 ;
;10825 ;WAIT.FOR.PREAMBLE1:
;10826 ;
;10827 ;        INCR.LPCNTR/ON,           ;WAIT FOR 8 MICROSECONDS
;10828 ;        BENO/CNTR.OVFLW,          ;TO REACH PREAMBLE.
;10829 ;        NAD/WAIT.FOR.PREAMBLE1
;10830 ;
;10831 ;------------------------------------------------------------------
;10832 ;=1
```

```
;10833 ;
;10834 ;SETUP.LOOP.LOCK1:
;10835 ;
;10836 ;       LOOP.LOCK/ON,           ;SETUP PHASE LOCK LOOP.
;10837 ;       FUNC/CLR.ECC,
;10838 ;       LOAD.LOOP.CNTR.[0E0],
;10839 ;       NAD/WAIT.LOOP.LOCK1
;10840 ;
;10841 ;-----------------------------------------------------------------------
;10842 ;=0
;10843 ;
;10844 ;WAIT.LOOP.LOCK1:
;10845 ;
;10846 ;       INCR.LPCNTR/ON,         ;ALLOW PHASE LOCK LOOP 4 BYTES
;10847 ;       LOOP.LOCK/ON,           ;OF PREAMBLE TO LOCK UP.
;10848 ;       UCON/CLR.MATCH,
;10849 ;       BENO/CNTR.OVFLW,
;10850 ;       NAD/WAIT.LOOP.LOCK1
;10851 ;
;10852 ;-----------------------------------------------------------------------
;10853 ;=1
;10854 ;
;10855 ;SETUP.RD.HDR1:
;10856 ;
;10857 ;       LOOP.LOCK/ON,           ;SETUP INITIAL CONDITIONS
;10858 ;       ENB.CLR.BC/ON,          ;FOR READING HEADER.
;10859 ;       FUNC/CLR.DSR,
;10860 ;       LOAD.LOOP.CNTR.[0F1]    ;256-15=241=F1
;10861 ;
;10862 ;-----------------------------------------------------------------------
;10863 ;
;10864 ;CHGCLK1A:
;10865 ;
;10866 ;       ENB.DSK.CLK,            ;THE MICROSEQUENCER WILL STALL
;10867 ;       LOOP.LOCK/ON,           ;UNTIL SYNC SEEN IS SET (SYNC BIT FOUND)
;10868 ;       ENB.CLR.BC/ON,          ;THEN WE USE CLOCK FROM DATA SEPARATOR.
;10869 ;       GEN.SYNC.[80]
;10870 ;
;10871 ;-----------------------------------------------------------------------
;10872 ;=0
;10873 ;
;10874 ;READ.HDR1:
;10875 ;
;10876 ;       LOOP.LOCK/ON,           ;SHIFT IN 15 MORE HEADER BITS AND
;10877 ;       DATA.FIELD/ON,          ;COMPARE WITH SERIAL DAR.
;10878 ;       ENB.CLR.BC/ON,
;10879 ;       INCR.LPCNTR/ON,
;10880 ;       BENO/CNTR.OVFLW,
;10881 ;       NAD/READ.HDR1
;10882 ;
;10883 ;-----------------------------------------------------------------------
;10884 ;=1
;10885 ;
;10886 ;TEST.HDR.MATCH1:
;10887 ;
```

```
;10888 ;        LOOP.LOCK/ON,           ;HEADER HAS BEEN READ.
;10889 ;        UCON/TEST.MATCH,        ;TURN ON TEST MATCH, OFF ENB CLR BC.
;10890 ;        DATA.FIELD/ON,
;10891 ;        NAD/BRANCH.HDR.MATCH1
;10892 ;
;10893 ;-----------------------------------------------------------------------
;10894 ;
;10895 ;BRANCH.HDR.MATCH1:
;10896 ;
;10897 ;        LOOP.LOCK/ON,
;10898 ;        DATA.FIELD/ON,          ;RESULTS OF HDR COMP NOW IN MISMATCH.
;10899 ;        BEN2/MISMATCH.L,
;10900 ;        NAD/HDR.COMP1
;10901 ;
;10902 ;-----------------------------------------------------------------------
;10903 ;=1**
;10904 ;
;10905 ;HDR.COMP1:
;10906 ;
;10907 ;        LOOP.LOCK/ON,           ;HEADER COMPARE!
;10908 ;        DATA.FIELD/ON,          ;CONTINUE TO READ THE HDR DENSITY WORD.
;10909 ;        LOAD.LOOP.CNTR.[0F2],   ;256-14=242=F2
;10910 ;        NAD/RD.DENSITY.WD1
;10911 ;
;10912 ;-----------------------------------------------------------------------
;10913 ;=0
;10914 ;
;10915 ;RD.DENSITY.WD1:
;10916 ;
;10917 ;        INCR.LPCNTR/ON,         ;CONT TO SHIFT IN THE HDR DENSITY WD.
;10918 ;        LOOP.LOCK/ON,
;10919 ;        DATA.FIELD/ON,
;10920 ;        BEN0/CNTR.OVFLW,
;10921 ;        NAD/RD.DENSITY.WD1
;10922 ;
;10923 ;-----------------------------------------------------------------------
;10924 ;=1
;10925 ;
;10926 ;CRC.PREP1:
;10927 ;
;10928 ;        LOAD.LOOP.CNTR.[0F1],   ;PREPARE TO READ THE CRC WORD.
;10929 ;        LOOP.LOCK/ON,           ;256-15=241=F1
;10930 ;        CHECK.FIELD/ON          ;TURN DFLD OFF AND CFLD ON.
;10931 ;
;10932 ;-----------------------------------------------------------------------
;10933 ;=0
;10934 ;
;10935 ;RD.CRC.WD1:
;10936 ;
;10937 ;        INCR.LPCNTR/ON,         ;NOW WE SHIFT IN THE CRC WORD
;10938 ;        LOOP.LOCK/ON,
;10939 ;        CHECK.FIELD/ON,
;10940 ;        BEN0/CNTR.OVFLW,
;10941 ;        NAD/RD.CRC.WD1
;10942 ;
```

```
;----------------------------------------------------------------------
;=1
;
;CHECK.CRC1:
;
;        LOOP.LOCK/ON,
;        UCON/CHECK.ECC            ;DROP CFLD. CLOCK ECC FLOP
;
;----------------------------------------------------------------------
;
;TEST.FOR.ERR1:
;
;        LOOP.LOCK/ON,             ;WAS THERE A CRC ERROR?
;        BEN1/ERROR.L
;
;----------------------------------------------------------------------
;=1*
;
;NOT.CRC.ERR1:
;
;        LOOP.LOCK/ON,
;        FUNC/CLR.ECC,             ;CLEAR DC007
;        BEN2/F2,                  ;BRANCH ON FUNCTION BITS INTO WRT CHK,
;        BEN1/F1,                  ;RD DATA OR WRT DATA FLOWS.
;        NAD/BRANCH.WC.CMD1
;
;----------------------------------------------------------------------
;=00*
;
;BRANCH.WC.CMD1:
;
;        LOOP.LOCK/ON,             ;CONTINUE VIA WRITE CHECK FLOWS.
;        BEN1/FIFO.OVFLW.L,
;        NAD/CHGCLK1B
;
;----------------------------------------------------------------------
;=0*
;
;CHGCLK1B:
;
;        LOOP.LOCK/ON,             ;EVERYTHING IS GO FOR READING DATA FROM
;        ENB.SYS.CLK,              ;THIS SECTOR. RUN OFF SYSTEM CLOCK THRU
;        LOAD.LOOP.CNTR.[OF5],     ;HEADER POSTAMBLE AND PAST SPLICE.
;        NAD/HDR.POST1
;
;----------------------------------------------------------------------
;=0
;
;HDR.POST1:
;
;        INCR.LPCNTR/ON,           ;SHIFT OVER HEADER POSTAMBLE
;        BENO/CNTR.OVFLW,          ;LOOP LOCK OFF
;        NAD/HDR.POST1
;
;----------------------------------------------------------------------
```

```
;10998 ;=1
;10999 ;
;11000 ;DATA.PRE1:
;11001 ;
;11002 ;        LOOP.LOCK/ON,            ;NOW WE ARE IN THE DATA PREAMBLE
;11003 ;        LOAD.LOOP.CNTR.[0E0]
;11004 ;
;11005 ;-------------------------------------------------------------------
;11006 ;=0
;11007 ;
;11008 ;DATA.PRE1A:
;11009 ;
;11010 ;        LOOP.LOCK/ON,            ;ALLOW PHASE LOCK LOOP 4 BYTES
;11011 ;        INCR.LPCNTR/ON,          ;OF PREAMBLE TO LOCK UP
;11012 ;        BEN0/CNTR.OVFLW,
;11013 ;        NAD/DATA.PRE1A
;11014 ;
;11015 ;-------------------------------------------------------------------
;11016 ;=1
;11017 ;
;11018 ;SETUP.RD.DATA1:
;11019 ;
;11020 ;        LOOP.LOCK/ON,            ;SETUP INITIAL CONDITIONS
;11021 ;        CLR.FIFO.ADR,            ;FOR READING DATA.
;11022 ;        UCON/CLR.MATCH,
;11023 ;        LOAD.LOOP.CNTR.[0FB]
;11024 ;
;11025 ;-------------------------------------------------------------------
;11026 ;
;11027 ;CHGCLK1C:
;11028 ;
;11029 ;        LOOP.LOCK/ON,            ;THE MICROSEQUENCER WILL STALL UNTIL
;11030 ;        ENB.DSK.CLK,             ;SYNC SEEN IS SET.
;11031 ;        GEN.SYNC.[80],           ;THEN WE USE THE DATA SEPARATOR CLK.
;11032 ;        WRT.CHK/ON,
;11033 ;        ENB.FIFO/ON
;11034 ;
;11035 ;-------------------------------------------------------------------
;11036 ;
;11037 ;INCR.FIFO.CNTR1:
;11038 ;
;11039 ;        LOOP.LOCK/ON,            ;ADVANCE FIFO CNTR TO REFLECT
;11040 ;        DATA.FIELD/ON,           ;HARDWARE LOAD.
;11041 ;        WRT.CHK/ON,
;11042 ;        INCR.FIFO.CNTR/ON        ;SHIFT IN 2ND BIT
;11043 ;
;11044 ;-------------------------------------------------------------------
;11045 ;=0
;11046 ;
;11047 ;READ.DATA1:
;11048 ;
;11049 ;        LOOP.LOCK/ON,            ;SHIFT IN 5 DATA BITS.
;11050 ;        DATA.FIELD/ON,
;11051 ;        WRT.CHK/ON,
;11052 ;        INCR.LPCNTR/ON,
;11053 ;        BENO/CNTR.OVFLW,
;11054 ;        NAD/READ.DATA1
;11055 ;
;11056 ;-----------------------------------------------------------------
;11057 ;=1
;11058 ;
;11059 ;LOAD.BYTE1:
;11060 ;
;11061 ;        LOOP.LOCK/ON,               ;8TH DATA BIT
;11062 ;        DATA.FIELD/ON,
;11063 ;        WRT.CHK/ON,
;11064 ;        ENB.FIFO/ON,
;11065 ;        LOAD.DSR/ON,                ;LOAD NEXT BYTE FROM FIFO INTO DSR.
;11066 ;        LOAD.LOOP.CNTR.[OF9],
;11067 ;        INCR.FIFO.CNTR/ON,
;11068 ;        NAD/READ.DATA1A
;11069 ;
;11070 ;-----------------------------------------------------------------
;11071 ;=10
;11072 ;
;11073 ;READ.DATA1A:
;11074 ;
;11075 ;        LOOP.LOCK/ON,               ;SHIFT IN 7 DATA BITS
;11076 ;        DATA.FIELD/ON,
;11077 ;        WRT.CHK/ON,
;11078 ;        INCR.LPCNTR/ON,
;11079 ;        BENO/CNTR.OVFLW,
;11080 ;        NAD/READ.DATA1A
;11081 ;
;11082 ;-----------------------------------------------------------------
;11083 ;=11
;11084 ;
;11085 ;LOAD.BYTE1A:
;11086 ;
;11087 ;        LOOP.LOCK/ON,               ;8TH DATA BIT
;11088 ;        DATA.FIELD/ON,
;11089 ;        ENB.FIFO/ON,
;11090 ;        WRT.CHK/ON,
;11091 ;        LOAD.DSR/ON,                ;LOAD NEXT BYTE FROM FIFO INTO DSR.
;11092 ;        LOAD.LOOP.CNTR.[OF9],
;11093 ;        INCR.FIFO.CNTR/ON,
;11094 ;        BEN1/FIFO.OVFLW.L,
;11095 ;        NAD/READ.DATA1A
;11096 ;
;11097 ;-----------------------------------------------------------------
;11098 ;=00
;11099 ;
;11100 ;RD.DATA.CRC1:
;11101 ;
;11102 ;        LOOP.LOCK/ON,               ;PREPARE TO READ THE CRC WORD.
;11103 ;        DATA.FIELD/ON,
;11104 ;        UCON/TEST.MATCH,            ;STROBE RESULTS OF DATA XOR.
;11105 ;        LOAD.LOOP.CNTR.[OFO],
;11106 ;        NAD/RD.DATA.CRC1A
;11107 ;
```

```
;11108 ;--------------------------------------------------------------------------
;11109 ;=0
;11110 ;
;11111 ;RD.DATA.CRC1A:
;11112 ;
;11113 ;        LOOP.LOCK/ON,
;11114 ;        CHECK.FIELD/ON,
;11115 ;        INCR.LPCNTR/ON,          ;SHIFT IN THE CRC WORD
;11116 ;        BENO/CNTR.OVFLW,
;11117 ;        NAD/RD.DATA.CRC1A
;11118 ;
;11119 ;--------------------------------------------------------------------------
;11120 ;=1
;11121 ;
;11122 ;CHECK.CRC1A:
;11123 ;
;11124 ;        LOOP.LOCK/ON,            ;DROP CFLD. CLOCK ECC FLOP
;11125 ;        UCON/CHECK.ECC
;11126 ;
;11127 ;--------------------------------------------------------------------------
;11128 ;
;11129 ;TEST.FOR.ERR1A:
;11130 ;
;11131 ;        ENB.CPU.CLK,             ;SWITCH OVER TO CPU CLOCK
;11132 ;        LOOP.LOCK/ON,            ;FOR CPU FAST INTERRUPT OR TERMINATE.
;11133 ;        BEN1/ERROR.L             ;WAS THERE A CRC OR WRT CHK ERROR?
;11134 ;
;11135 ;--------------------------------------------------------------------------
;11136 ;=1*
;11137 ;
;11138 ;NOT.CRC.ERR1A:
;11139 ;
;11140 ;        CLR.FIFO.ADR,            ;CLEAR THE FIFO ADDRESS COUNTER
;11141 ;        UCMD/SET.XFER.RQST,
;11142 ;        NAD/WAIT.XFER.GRANT1
;11143 ;
;11144 ;--------------------------------------------------------------------------
;11145 ;=1*
;11146 ;
;11147 ;WAIT.XFER.GRANT1:
;11148 ;
;11149 ;        BEN1/XFER.REQ,           ;CPU TURNS ON CRDY IF LAST SECTOR
;11150 ;        NAD/WAIT.XFER.GRANT1     ;   THEN SENDS GRANT
;11151 ;
;11152 ;--------------------------------------------------------------------------
;11153 ;=0*
;11154 ;
;11155 ;TEST.XFER.DONE1:
;11156 ;
;11157 ;        UCMD/CHG.FIFO,           ;SWITCH FIFOS, IN CASE THERE'S MORE DATA
;11158 ;        BENO/CRDY.L,
;11159 ;        NAD/XFER.DONE1
;11160 ;
;11161 ;--------------------------------------------------------------------------
;11162 ;=0
```

```
;11163 ;
;11164 ;XFER.DONE1:
;11165 ;
;11166 ;        UCMD/SET.CRDY,
;11167 ;        INCR.DAR/ON,
;11168 ;        FUNC/SET.INT,              ;CPU SIGNALS END OF TRANSFER.
;11169 ;        NAD/SET.INT.AND.IDLE
;11170 ;
;11171 ;----------------------------------------------------------------------
;11172 ;=111111110
;11173 ;
;11174 ;SET.INT.AND.IDLE:
;11175 ;
;11176 ;        FUNC/SET.INT,              ; The purpose of this state is to give CRDY one
;11177 ;        NAD/IDLE                   ; state to reset before asserting interrupt.
;11178 ;                                   ; CRDY is now used in the INT equation.
;11179 ;
;11180 ;----------------------------------------------------------------------
;11181 ;
;11182 ;=1
;11183 ;
;11184 ;RTN.TO.IDLE:
;11185 ;IDLE:
;11186 ;
;11187 ;        BENO/CRDY.L,               ;LOOK FOR GO (CRDY L)
;11188 ;        NAD/SETUP.DRV.CHK
;11189 ;
;11190 ;----------------------------------------------------------------------
;11191 ;=0
;11192 ;
;11193 ;SETUP.DRV.CHK:
;11194 ;
;11195 ;        LOAD.LOOP.CNTR.[OFB],      ;COUNTER WILL BE USED
;11196 ;        NAD/SEL.NXT.DRV            ;TO SEQUENCE THRU DRIVES.
;11197 ;
;11198 ;----------------------------------------------------------------------
;11199 ;
;11200 ;SEL.NXT.DRV:
;11201 ;
;11202 ;        SEL.DRV/UPROG,
;11203 ;        BENO/CNTR.OVFLW,
;11204 ;        NAD/CHK.DRV.STATUS
;11205 ;
;11206 ;----------------------------------------------------------------------
;11207 ;=0
;11208 ;
;11209 ;CHK.DRV.STATUS:
;11210 ;        ; Give drive status time to settle down.
;11211 ;
;11212 ;        SEL.DRV/UPROG,             ;SELECT DRIVE
;11213 ;        NAD/DRV.STATUS.DISPATCH
;11214 ;
;11215 ;----------------------------------------------------------------------
;11216 ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
;11217 ;
```

```
                  ;11218 ;DRV.STATUS.DISPATCH:
                  ;11219 ;        SEL.DRV/UPROG,          ;SELECT DRIVE
                  ;11220 ;        BEN1/DRIVE.ONLINE,      ;AND BRANCH ON STATUS.
                  ;11221 ;        BENO/DRIVE.RDY,
                  ;11222 ;        NAD/-ERR.-OL.-RDY
                  ;11223 ;
                  ;11224 ;
                  ;11225 ;-----------------------------------------------------------------
                  ;11226 ;=011
                  ;11227 ;
                  ;11228 ;-ERR.OL.RDY:
                  ;11229 ;
                  ;11230 ;        SEL.DRV/UPROG,          ;NOP
                  ;11231 ;        INCR.LPCNTR/ON,         ;ADD 1 TO DRIVE SELECT.
                  ;11232 ;        NAD/SEL.NXT.DRV
                  ;11233 ;
                  ;11234 ;-----------------------------------------------------------------
                  ;11235 ;
                  ;11236 ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                  ;11237 ;
                  ;11238 ;               UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                  ;11239 ;
                  ;11240 ;-----------------------------------------------------------------
                  ;11241 ;
                  ;11242 ;=1
                  ;11243 ;
                  ;11244 ;RTN.TO.IDLE:
                  ;11245 ;IDLE:
                  ;11246 ;
                  ;11247 ;        BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
                  ;11248 ;        NAD/SETUP.DRV.CHK
                  ;11249 ;
                  ;11250 ;-----------------------------------------------------------------
                  ;11251
                  ;11252 ;Set up parameters for time-out error report :
                  ;11253
U 09D9, 90,9015   ;11254          MISC [SET.PORT.I.O]                     ;select the port to bus
U 09DA, 97,8015   ;11255          PORT.READ PORT.ADRS[CSR]                ;read CSR back
U 09DB, 3B,3215   ;11256          MOV PORT TO LS[PORTO]                   ; from the port and
U 09DC, 37,3395   ;11257          MOV LS[PORTO] TO WR[3]                  ;  load it in WR3
                  ;11258
U 09DD, 36,4095   ;11259          MOV LS[#1] TO WR[1]
U 09DE, 3E,8295   ;11260          MOV WR[1] TO LS[ERROR.NUMBER]           ;set up error number = 1 in LS
U 09DF, E5,8A15   ;11261          CLR LS[ERROR.MASK]                      ;no error mask used here
U 09E0, 36,4A95   ;11262          MOV LS[IDC] TO WR[1]                    ;store module code in LS to
U 09E1, BE,8C95   ;11263          MOV WR[1] TO LS[MODULE.NUM]             ; modules
U 09E2, 36,6E95   ;11264          MOV LS[NA.EXP.REC] TO WR[1]             ;put "N/A" for printout data
U 09E3, 47,7095   ;11265          BIS LS[OTHER.DATA] TO WR[1]             ;put CSR under "OTHER DATA"
U 09E4, BE,8095   ;11266          MOV WR[1] TO LS[CONTROL.STATUS]
                  ;11267
U 09E5, 2F,8015   ;11268          CLR WR[0]                               ;Initialize time-out counter
U 09E6, BF,8C15   ;11269          MOV WR[0] TO LS[TIMEOUT.CNTR]
                  ;11270
                  ;11271 ;Timeout loop to wait for XFER REQ after write check has been issued :
                  ;11272
```

```
                   ;11273 WAIT.IDC.T8.2:
U 09E7, 37,8C15    ;11274          MOV LS[TIMEOUT.CNTR] TO WR[0]             ;load the current time count
U 09E8, 37,9095    ;11275          MOV LS[WAIT.TIME1] TO WR[1]               ;time limit on write check
                   ;11276          CMP WR[0] WITH WR[1],                     ;timeout yet ???
U 09E9, 26,4085    ;11277           DT(LONG)&SET.ALU.CC
U 09EA, 08,9F09    ;11278          JMP.IF[EQL] TO [T8.ERR.1]                 ;yes, timeout on write check
U 09EB, 20,4015    ;11279          INC WR[0]                                 ;inc. time count
U 09EC, BF,8C15    ;11280          MOV WR[0] TO LS[TIMEOUT.CNTR]
U 09ED, DB,001A    ;11281          SKIP.IF[NO.FAST.INTERRUPT]                ;check if XFER REQ is set
U 09EE, 88,9F64    ;11282          JMP [IDC.DONE.T8.2]                       ;yes, XFER REQ is set now
U 09EF, 88,9E74    ;11283          JMP [WAIT.IDC.T8.2]                       ;no, wait some more
                   ;11284
                   ;11285 ;Force an error in the CHECK.RESULT routine for this timout :
                   ;11286
                   ;11287 T8.ERR.1:
U 09F0, 2F,8015    ;11288          CLR WR[0]                                 ;set unequal values
U 09F1, 36,4095    ;11289          MOV LS[#1] TO WR[1]                       ; between WR0 and WR1
U 09F2, BE,8995    ;11290          MOV WR[3] TO LS[ADDRESS.DATA]             ;value for "OTHER DATA"
U 09F3, 08,693C    ;11291          JSR [CHECK.RESULT]                        ;force an error
U 09F4, 88,9CC4    ;11292          JMP [T.8.4]                               ;err and if LOE is set
U 09F5, 88,9F64    ;11293          JMP [IDC.DONE.T8.2]
                   ;11294
                   ;11295 IDC.DONE.T8.2:
U 09F6, 90,FA15    ;11296          MISC2 [TRANSFER.GRANT]                    ;send grant to clr XFER REQ
U 09F7, 08,9F84    ;11297          JMP [END.T8]                              ;test ends
                   ;11298
                   ;11299 END.T8:                                            ;end of test
```

```
;11300 .PAGE  "Test 9  R80 GET STATUS FUNCTION TEST (not run in CRD mode)"
;11301 ;++
;11302 ;
;11303 ; Test description:
;11304 ;
;11305 ;        This test will look at LS[DR0.R80] to determine if drive 0 has a R80.
;11306 ;        If LS[DR0.R80] is zero indicating no R80 in drive 0. The test will be
;11307 ;        skipped. If non-zero, the test will select the R80 drive and a get
;11308 ;        status function by writing to the CSR <2>. Then the test will wait for
;11309 ;        CRDY set and read the upper disk status byte FIFO A and compare that
;11310 ;        with the status expected.
;11311 ;
;11312 ;
;11313 ;                R80
;11314 ;                status
;11315 ;                bits          functions         expected states
;11316 ;                -------       ---------         ---------------
;11317 ;
;11318 ;                 <0:4>        sector count      X random sector adr.
;11319 ;                                                  (will be masked out)
;11320 ;                 <5:7>        not used          X don't care
;11321 ;                                                  (will be masked out)
;11322 ;                   <8>        fault             0 no fault in the drive
;11323 ;                   <9>        plug valid        1 address plug installed
;11324 ;                  <10>        seek error        0 not doing a seek
;11325 ;                  <11>        on cylinder       1 heads on the disk
;11326 ;                  <12>        drive ready       1 drive ready for commands
;11327 ;                  <13>        write protect     0 write enabled
;11328 ;               <14:15>        not used          X don't care
;11329 ;                                                  (will be masked out)
;11330 ;
;11331 ; Assumption:
;11332 ;
;11333 ;        Both CPU micros and IDC part I has been run successfully and
;11334 ;        drive 0 is a R80, up and spinning and ready to accept commands.
;11335 ;
;11336 ;        This test assumes "DISK DRIVE SIZING TEST" has been run prior to this
;11337 ;        test.
;11338 ;
;11339 ; Test steps:
;11340 ;
;11341 ;          1. Load LS[DR0.R80] and examine its content
;11342 ;          2. If value is zero, go to end of test
;11343 ;          3. If value is non-zero, write CSR <2> for get status function
;11344 ;          4. Set up a time out loop waiting for CRDY to set
;11345 ;          5. If timeout on get status command;
;11346 ;          6.      Then set up error parameters and report ERROR 1 and
;11347 ;                   force an error in JSR[CHECK.RESULT]
;11348 ;          7. Else port select FIFO A
;11349 ;          8. Clear FIFO A address counter
;11350 ;          9. Read the 2nd byte of the status from FIFO A
;11351 ;         10. Set up all pertinent error parameters for ERROR 2
;11352 ;         11. Load the error mask for R80 status from LS[#FFFFC0FF]
;11353 ;         12. Load the expected status from LS[#1A00]
;11354 ;         13. Use JSR[CHECK.RESULT] to check out the status
```

```
                  ;       14. If status not compare, report ERROR 2
                  ;       15. Else test ends
                  ;
                  ; Errors:
                  ;
                  ; ERROR 1 - Time out waiting for CRDY to set after a Get Status
                  ;           command has been issued to the IDC
                  ;
                  ;       Printout:
                  ;             EXPECTED        RECEIVED        MODULE
                  ;             N/A             N/A             R80
                  ;
                  ; ERROR 2 - R80 drive status error
                  ;
                  ;       Printout:
                  ;             EXPECTED        RECEIVED        MODULE
                  ;             R80 status      R80 status      R80
                  ;
                  ; Debug:
                  ;
                  ;   ERROR 1: The microcontroller selects FIFO A and clears its address counters
                  ;            and the 16 bits status of the R80 disk drive is asserted to the
                  ;            IDC in parallel format. R80 status and sector counts are applied
                  ;            to the R80 MUX. After FIFO A has been selected and its address
                  ;            counter cleared, the microcontroller enables the R80 SECTOR COUNT
                  ;            through the R80 MUX, writes it into FIFO A and increments the
                  ;            address counter. Then the microcontroller enables the R80 STATUS
                  ;            through the R80 MUX and writes it into FIFO A. When timeout occurs
                  ;            check the R80 mux for its USEL R80 ST, UENB R80 ST and R80 STATUS
                  ;            signals for faults.
                  ;
                  ;   ERROR 2: Refer to the expected operational R80 status as listed above, and
                  ;            if error occurs despite a correct status of the R80; check the
                  ;            two 74LS257 R80 multiplexers and every interface signal that goes
                  ;            between them and the DR SEL LOG PAL 16L8.
                  ;
                  ;--

                  T.9:
U 09F8, B6,5E15             MOV LS[BEGIN.TEST] TO WR[0]             ;set bit 15 in WR[0] for
                                                                    ; control and status word
U 09F9, 3E,8015             MOV WR[0] TO LS[CONTROL.STATUS]         ;set bit 15 in control and
                                                                    ; status word - bit 15
                                                                    ; indicates start of test to
                                                                    ; console
U 09FA, 10,E015             MISC [SET.CP.ATTN]                      ;issue CPU ATTN to console

                  WAIT.T9.0:
U 09FB, 08,9FB4             JMP [WAIT.T9.0]                         ;loop here waiting for
                                                                    ; console to respond

U 09FC, 65,A015             CLR LS[PREVIOUS.ERROR]                  ;clear previous error number
                                                                    ; in LS
U 09FD, 17,CC15             PORT.CONTROL [CLEAR.IDC]                ;init. IDC
U 09FE, 17,CC15             PORT.CONTROL [CLEAR.IDC]
```

```
U 09FF, 17,CC15   ;11410          PORT.CONTROL [CLEAR.IDC]
                  ;11411
                  ;11412 T.9.1:
U 0A00, B7,0C15   ;11413          MOV LS[DRO.R80] TO WR[0]                ;load the R80 presence indicator
                  ;11414          TST WR[0],                              ;see if drive zero has an R80
U 0A01, 20,0035   ;11415           DT(LONG)&SET.ALU.CC
U 0A02, 88,A2F9   ;11416          JMP.IF[EQL] TO [END.T9]                 ;no R80, go to end of test
                  ;11417
                  ;11418 T.9.2:
U 0A03, 37,0215   ;11419          MOV LS[GET.STATUS] TO WR[0]             ;fetch the get status command
U 0A04, 47,9C15   ;11420          BIS LS[SETCSR.DS0] TO WR[0]             ;select drive 0
U 0A05, 17,A015   ;11421          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;write to the CSR
                  ;11422
                  ;11423 ;*** R80 GET STATUS IDC FUNCTION CODES : ***
                  ;11424
                  ;11425 ;-----------------------------------------------------------------------
                  ;11426 ;=1
                  ;11427 ;
                  ;11428 ;TEST.OR.RUN:
                  ;11429 ;
                  ;11430 ;        BEN2/MAINT,              ;IS THAT YOU, VICKIE?
                  ;11431 ;        NAD/DRIVE.TYPE
                  ;11432 ;
                  ;11433 ;-----------------------------------------------------------------------
                  ;11434 ;=0**
                  ;11435 ;
                  ;11436 ;DRIVE.TYPE:
                  ;11437 ;
                  ;11438 ;        BEN0/R80,                ;WHICH TYPE OF DRIVE WAS SELECTED?
                  ;11439 ;        NAD/RL02.IFORK
                  ;11440 ;
                  ;11441 ;-----------------------------------------------------------------------
                  ;11442 ;=1
                  ;11443 ;
                  ;11444 ;R80.IFORK:
                  ;11445 ;
                  ;11446 ;        UCMD/SET.FIFOA,
                  ;11447 ;        FUNC/CLR.ECC.ERROR,      ; Enable clear of ECC/CRC errors for new function
                  ;11448 ;        BEN0/F0,                 ;8 WAY INSTRUCTION FORK.
                  ;11449 ;        BEN1/F1,                 ;BRANCH ON F<2:0> IN THE CSR.
                  ;11450 ;        BEN2/F2,
                  ;11451 ;        NAD/#NOP.CMD
                  ;11452 ;
                  ;11453 ;-----------------------------------------------------------------------
                  ;11454 ;=010
                  ;11455 ;
                  ;11456 ;#GET.STAT.CMD:
                  ;11457 ;
                  ;11458 ;        NAD/#SETUP.FIFO2         ;START OF THE R80 GET STATUS COMMAND
                  ;11459 ;
                  ;11460 ;-----------------------------------------------------------------------
                  ;11461 ;
                  ;11462 ;#SETUP.FIFO2:
                  ;11463 ;
                  ;11464 ;        UCMD/SET.FIFOA, ;SELECT FIFOA
```

```
;11465 ;        CLR.FIFO.ADR                ;CLEAR THE ADDRESS COUNTER
;11466 ;
;11467 ;-------------------------------------------------------------------------
;11468 ;
;11469 ;#LOAD.LOBYTE2:
;11470 ;
;11471 ;        ENB.R80.STAT.LOBYTE,        ;TRISTATE ENABLE ONTO BUS IN.
;11472 ;        WRT.FIFO/ON,
;11473 ;        INCR.FIFO.CNTR/ON           ;WRITE FIFO AND ADVANCE ADRS COUNTER.
;11474 ;
;11475 ;-------------------------------------------------------------------------
;11476 ;
;11477 ;#LOAD.HIBYTE2:
;11478 ;
;11479 ;        ENB.R80.STAT.HIBYTE,        ;SAME OPR AS LOBYTE
;11480 ;        WRT.FIFO/ON
;11481 ;
;11482 ;-------------------------------------------------------------------------
;11483 ;
;11484 ;#END.STAT.CMD2:
;11485 ;
;11486 ;        CLR.FIFO.ADR,               ;INITIALIZE CNTR FOR PORT READ
;11487 ;        NAD/#XFER.DONE4
;11488 ;
;11489 ;-------------------------------------------------------------------------
;11490 ;
;11491 ;#XFER.DONE4:
;11492 ;
;11493 ;        UCMD/SET.CRDY,              ;SIGNAL END OF TRANSFER
;11494 ;        FUNC/SET.INT,
;11495 ;        NAD/SET.INT.AND.IDLE
;11496 ;
;11497 ;-------------------------------------------------------------------------
;11498 ;=111111110
;11499 ;
;11500 ;SET.INT.AND.IDLE:
;11501 ;
;11502 ;        FUNC/SET.INT,               ; The purpose of this state is to give CRDY one
;11503 ;        NAD/IDLE                    ; state to reset before asserting interrupt.
;11504 ;                                    ; CRDY is now used in the INT equation.
;11505 ;
;11506 ;-------------------------------------------------------------------------
;11507 ;
;11508 ;=1
;11509 ;
;11510 ;RTN.TO.IDLE:
;11511 ;IDLE:
;11512 ;
;11513 ;        BENO/CRDY.L,                ;LOOK FOR GO (CRDY L)
;11514 ;        NAD/SETUP.DRV.CHK
;11515 ;
;11516 ;-------------------------------------------------------------------------
;11517 ;=0
;11518 ;
;11519 ;SETUP.DRV.CHK:
```

```
;
;       LOAD.LOOP.CNTR.[OFB],   ;COUNTER WILL BE USED
;       NAD/SEL.NXT.DRV         ;TO SEQUENCE THRU DRIVES.
;
;------------------------------------------------------------------
;
;SEL.NXT.DRV:
;
;       SEL.DRV/UPROG,
;       BENO/CNTR.OVFLW,
;       NAD/CHK.DRV.STATUS
;
;------------------------------------------------------------------
;=0
;
;CHK.DRV.STATUS:
;       ; Give drive status time to settle down.
;
;       SEL.DRV/UPROG,          ;SELECT DRIVE
;       NAD/DRV.STATUS.DISPATCH
;
;------------------------------------------------------------------
; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
;
;DRV.STATUS.DISPATCH:
;       SEL.DRV/UPROG,          ;SELECT DRIVE
;       BEN1/DRIVE.ONLINE,      ;AND BRANCH ON STATUS.
;       BENO/DRIVE.RDY,
;       NAD/-ERR.-OL.-RDY
;
;
;------------------------------------------------------------------
;=011
;
;-ERR.OL.RDY:
;
;       SEL.DRV/UPROG,          ;NOP
;       INCR.LPCNTR/ON,         ;ADD 1 TO DRIVE SELECT.
;       NAD/SEL.NXT.DRV
;
;------------------------------------------------------------------
;
; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
;
;               UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
;
;------------------------------------------------------------------
;
;=1
;
;RTN.TO.IDLE:
;IDLE:
;
;       BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
;       NAD/SETUP.DRV.CHK
```

```
                  ;
                  ;---------------------------------------------------------------------------------------


                  ;Set up parameters for time-out error report :

U 0A06, 36,4095           MOV LS[#1] TO WR[1]
U 0A07, 3E,8295           MOV WR[1] TO LS[ERROR.NUMBER]            ;set up error number = 1 in LS
U 0A08, E5,8A15           CLR LS[ERROR.MASK]                       ;no error used here
U 0A09, 36,4A95           MOV LS[IDC] TO WR[1]                     ;store module code in LS to
U 0A0A, BE,8C95           MOV WR[1] TO LS[MODULE.NUM]              ; modules
U 0A0B, 36,6E95           MOV LS[NA.EXP.REC] TO WR[1]              ;put "N/A" for printout data
U 0A0C, BE,8095           MOV WR[1] TO LS[CONTROL.STATUS]

U 0A0D, 2F,8015           CLR WR[0]                                ;Initialize time-out counter
U 0A0E, BF,8C15           MOV WR[0] TO LS[TIMEOUT.CNTR]

                  WAIT.IDC.T9.1:
U 0A0F, 37,8C15           MOV LS[TIMEOUT.CNTR] TO WR[0]            ;load the current time count
U 0A10, 37,9095           MOV LS[WAIT.TIME1] TO WR[1]              ;time limit on get status
                          CMP WR[0] WITH WR[1],                    ;timeout yet ???
U 0A11, 26,40B5            DT(LONG)&SET.ALU.CC
U 0A12, 08,A189           JMP.IF[EQL] TO [T9.ERR.1]                ;yes, timeout on get status
U 0A13, 20,4015           INC WR[0]                                ;inc. time count
U 0A14, BF,8C15           MOV WR[0] TO LS[TIMEOUT.CNTR]
U 0A15, 88,700C           JSR [WAIT.CSR.CRDY]                      ;check if CRDY is set
U 0A16, 88,A1D4           JMP [IDC.DONE.T9.1]                      ;yes, CRDY is set now
U 0A17, 88,A0F4           JMP [WAIT.IDC.T9.1]                      ;no, wait some more

                  ;Force an error in the CHECK.RESULT routine for this timeout :

                  T9.ERR.1:
U 0A18, 2F,8015           CLR WR[0]                                ;set unequal values
U 0A19, 36,4095           MOV LS[#1] TO WR[1]                      ; between WR0 and WR1
U 0A1A, 08,693C           JSR [CHECK.RESULT]                       ;force an error
U 0A1B, 88,A034           JMP [T.9.2]                              ;err and if LOE is set
U 0A1C, 88,A1D4           JMP [IDC.DONE.T9.1]

                  ;Set up error parameters for ERROR 2 report :

                  IDC.DONE.T9.1:
U 0A1D, 36,4215           MOV LS[#2] TO WR[0]                      ;error number = 2
U 0A1E, BE,8215           MOV WR[0] TO LS[ERROR.NUMBER]
U 0A1F, 37,BC15           MOV LS[#FFFFC0FF] TO WR[0]               ;WR0 = #FFFFC0FF is the
U 0A20, 3E,8A15           MOV WR[0] TO LS[ERROR.MASK]              ; error mask for R80 status
U 0A21, 36,4A95           MOV LS[IDC] TO WR[1]                     ;identify IDC & R80
U 0A22, C7,4E95           BIS LS[R80] TO WR[1]                     ; as module codes to indicate
U 0A23, BE,8C95           MOV WR[1] TO LS[MODULE.NUM]              ;  modules at fault
U 0A24, E5,8015           CLR LS[CONTROL.STATUS]                   ;clear LS[NA.EXP.REC]

                  ;Read the R80 disk status from the 2nd byte of FIFO A :

U 0A25, 17,D815           PORT.CONTROL [SEL.FIFO.A]                ;read FIFO A
U 0A26, 17,C015           PORT.CONTROL [CLEAR.FIFO.CNTR]           ;clear FIFO A address
U 0A27, 90,9015           MISC [SET.PORT.I.O]                      ;select IDC
```

```
U 0A28, 97,8C15  ;11630          PORT.READ PORT.ADRS[DATA.WORD]              ;read R80 status from FIFO A
U 0A29, 3B,3215  ;11631          MOV PORT TO LS[PORTO]                       ;save it temporarily
U 0A2A, 37,3215  ;11632          MOV LS[PORTO] TO WR[0]                      ;retrieve it as received status
U 0A2B, 37,BE95  ;11633          MOV LS[#1A00] TO WR[1]                      ;fetch the expected status
U 0A2C, 08,693C  ;11634          JSR [CHECK.RESULT]                          ;correct status ???
U 0A2D, 88,A034  ;11635          JMP [T.9.2]                                 ;err and if LOE is set
U 0A2E, 08,A2F4  ;11636          JMP [END.T9]                                ;test ends
                 ;11637
                 ;11638 END.T9:                                              ;end of test 9
                 ;11639
```

```
.PAGE  "Test A  R80 SECTOR COUNT SIGNAL TEST (not run in CRD mode)"
;++
;
; Test Description:
;
;       The SECTOR COUNT signals will be monitored by asserting get status
;       commands to the R80 and check the sector count patterns within the
;       status words that echo back in the data buffer. To monitor the sector
;       count signals in a controllable fashion, get status functions will be
;       issued within a given period of time and each sector count bit will
;       be checked to see if it changes state.
;
; Assumption:
;
;       Both CPU micros and IDC part I has been run successfully and
;       drive 0 is a R80, up and spinning and ready to accept commands;
;       R80 get status function is working.
;
;       This test assumes "DISK DRIVE SIZING TEST" has been run prior to this
;       test.
;
; Test steps:
;
;        1. Load LS[DR0.R80] and examine its content
;        2. If value is zero, go to end of test
;        3. Initialize the error mask by its last bit
;        4. Initialize a check-for-zero loop counter and zero-or-one flag
;        5. Else write CSR <2> for get status function
;        6. Wait for CRDY to set
;        7. Port select FIFO A
;        8. Clear FIFO A address counter
;        9. Read the 1st byte of sector count from FIFO A and store it up
;       10. See if it is going to check for zero-state of this sector count bit
;       11. If so,
;       12.     Then check if this sector count bit is zero, if so;
;       13.          Then set zero-or-one flag, init. check-for-one loop counter
;                         and go to step 5
;       14.          Else check if check-for-zero loop counter expires, if so;
;       15.               Then report ERROR 1
;       16.               Else inc. check-for-zero counter, go to step 5
;       17.     Else check if this sector count bit is one, if so
;       18.          Then check if all sector count bits done, if so
;       19.               Then go to step 24
;       20.               Else reset zero-or-one flag, rotate ERROR.MASK left 1
;                               bit and go to step 4
;       21.          Else check if check-for-one loop counter expires, if so;
;       22.                  Then report ERROR 2
;       23.                  Else inc. check-for-one counter, go to step 5
;       24. Test ends
;
; Error:
;
; ERROR 1 - Time out waiting for sector count bit to be in zero-state
;
; ERROR 2 - Time out waiting for sector count bit to be in one-state
```

```
                   ;11695 ;
                   ;11696 ;         Printout:
                   ;11697 ;                 EXPECTED              RECEIVED              MODULE
                   ;11698 ;                 sector count          sector count          R80
                   ;11699 ;                 bit to flip           bit to flip
                   ;11700 ;
                   ;11701 ; Debug:
                   ;11702 ;
                   ;11703 ;    ERROR 1 & 2 : The sector expected might not be picked up by coincidence
                   ;11704 ;                  (but don't expect this to ever happen because the chance
                   ;11705 ;                   is so rare) or the sector count signals on the IDC are at
                   ;11706 ;                   fault; check the two 74LS257 R80 multiplexers and every
                   ;11707 ;                   interface signal that goes between them.
                   ;11708 ;
                   ;11709 ;--
                   ;11710
                   ;11711 T.A:
U 0A2F, B6,5E15    ;11712          MOV LS[BEGIN.TEST] TO WR[0]               ;set bit 15 in WR[0] for
                   ;11713                                                    ; control and status word
U 0A30, 3E,8015    ;11714          MOV WR[0] TO LS[CONTROL.STATUS]           ;set bit 15 in control and
                   ;11715                                                    ; status word - bit 15
                   ;11716                                                    ; indicates start of test to
                   ;11717                                                    ; console
U 0A31, 10,E015    ;11718          MISC [SET.CP.ATTN]                        ;issue CPU ATTN to console
                   ;11719
                   ;11720 WAIT.TA.0:
U 0A32, 08,A324    ;11721          JMP [WAIT.TA.0]                           ;loop here waiting for
                   ;11722                                                    ; console to respond
                   ;11723
U 0A33, 65,A015    ;11724          CLR LS[PREVIOUS.ERROR]                    ;clear previous error number
                   ;11725                                                    ; in LS
U 0A34, 17,CC15    ;11726          PORT.CONTROL [CLEAR.IDC]                  ;init. IDC
U 0A35, 17,CC15    ;11727          PORT.CONTROL [CLEAR.IDC]
U 0A36, 17,CC15    ;11728          PORT.CONTROL [CLEAR.IDC]
                   ;11729
                   ;11730 T.A.1:
U 0A37, B7,0C15    ;11731          MOV LS[DR0.R80] TO WR[0]                  ;load the R80 presence indicator
                   ;11732          TST WR[0],                                ;see if drive one has an R80
U 0A38, 20,0035    ;11733           DT(LONG)&SET.ALU.CC
U 0A39, 08,A7D9    ;11734          JMP.IF[EQL] TO [END.TA]                   ;no R80, go to end of test
                   ;11735
                   ;11736 ;Initialize error mask for JSR [CHECK.RESULT] :
                   ;11737
U 0A3A, B7,B815    ;11738          MOV LS[#FFFFFFFE] TO WR[0]                ;error mask for checking
U 0A3B, 3E,8A15    ;11739          MOV WR[0] TO LS[ERROR.MASK]               ; 1 sector count bit
                   ;11740
                   ;11741 ;Initialize check-for-zero loop counter and zero-or-one flag:
                   ;11742
                   ;11743 T.A.2:
U 0A3C, E5,1015    ;11744          CLR LS[T8]                                ;check-for-zero loop counter
U 0A3D, 65,1415    ;11745          CLR LS[T10]                               ;zero-or-one flag
                   ;11746
                   ;11747 ;Start a get status command, only the sector count part of the status word is used in this test :
                   ;11748
                   ;11749 T.A.3:
```

```
U 0A3E, 37,0215  ;11750          MOV LS[GET.STATUS] TO WR[0]                 ;fetch get status command
U 0A3F, 47,9C15  ;11751          BIS LS[SETCSR.DS0] TO WR[0]                 ;select drive 0
U 0A40, 17,A015  ;11752          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]        ;issue get status command
                 ;11753
                 ;11754          ;*********************************************************************
                 ;11755          ;* REFER TO THE R80 GET STATUS IDC FUNCTION CODES AS LISTED IN TEST 9 *
                 ;11756          ;*********************************************************************
                 ;11757
                 ;11758 WAIT.IDC.TA.1:
U 0A41, 88,700C  ;11759          JSR [WAIT.CSR.CRDY]                         ;check if CRDY is set
U 0A42, 88,A444  ;11760          JMP [IDC.DONE.TA.1]                         ;CRDY is set
U 0A43, 88,A414  ;11761          JMP [WAIT.IDC.TA.1]                         ;wait some more
                 ;11762
                 ;11763 ;Read the 1st byte (sector count) from FIFO A :
                 ;11764
                 ;11765 IDC.DONE.TA.1:
U 0A44, 17,D815  ;11766          PORT.CONTROL [SEL.FIFO.A]                   ;read FIFO A
U 0A45, 17,C015  ;11767          PORT.CONTROL [CLEAR.FIFO.CNTR]              ;clear FIFO A address
U 0A46, 90,9015  ;11768          MISC [SET.PORT.I.0]                         ;select IDC
U 0A47, 17,8815  ;11769          PORT.READ PORT.ADRS[DATA.BYTE]              ;1st byte = R80 sector count
U 0A48, 3B,3215  ;11770          MOV PORT TO LS[PORT0]                       ;save it temporarily
                 ;11771
                 ;11772 ;Decide if zero-state or one-state of the sector count bit <4:0> is being checked for :
                 ;11773
U 0A49, 36,1415  ;11774          MOV LS[T10] TO WR[0]                        ;fetch the zero-or-one flag
                 ;11775          TST WR[0],                                  ;zero-state checked already ??
U 0A4A, 20,0035  ;11776           DT(LONG)&SET.ALU.CC
U 0A4B, 88,A589  ;11777          JMP.IF[EQL] TO [CHECK.ZERO.STATE]           ;no, check for zero-state first
                 ;11778
                 ;11779 ;If zero-state has already been checked, check for one-state of this sector count bit :
                 ;11780
U 0A4C, 36,8A95  ;11781          MOV LS[ERROR.MASK] TO WR[1]                 ;current error mask
U 0A4D, A0,C295  ;11782          COM WR[1]                                   ;flip it
U 0A4E, 37,3215  ;11783          MOV LS[PORT0] TO WR[0]                      ;received sector count bit
                 ;11784          AND WR[1] TO WR[0],                         ;mask it out to check for one-state
U 0A4F, AF,C235  ;11785           DT(LONG)&SET.ALU.CC
U 0A50, 08,A669  ;11786          JMP.IF[EQL] TO [CHECK.TIMEOUT.1]            ;not one, see if time loop expires
                 ;11787
                 ;11788 ;If the sector count bit is a one, reset the zero-or-one flag and fix the error
                 ;11789 ; mask to check for the next higher bit. But first see if all sector count bits
                 ;11790 ; are checked; if so test done !
                 ;11791
U 0A51, B6,8A15  ;11792          MOV LS[ERROR.MASK] TO WR[0]                 ;fetch the current error mask
U 0A52, B7,BA95  ;11793          MOV LS[#FFFFFFEF] TO WR[1]                  ;value sector count bits all checked
                 ;11794          CMP WR[0] WITH WR[1],                       ;all done now ??
U 0A53, 26,4085  ;11795           DT(LONG)&SET.ALU.CC
U 0A54, 08,A7D9  ;11796          JMP.IF[EQL] TO [END.TA]                     ;yes, test ends
                 ;11797
U 0A55, A3,C015  ;11798          ROL WR[0]                                   ;rotate error mask to next higher bit
U 0A56, 3E,8A15  ;11799          MOV WR[0] TO LS[ERROR.MASK]                 ;save it for next pass
U 0A57, 88,A3C4  ;11800          JMP [T.A.2]
                 ;11801
                 ;11802 ;Check for zero-state first:
                 ;11803
                 ;11804 CHECK.ZERO.STATE:
```

```
U 0A58, 36,8A95   ;11805          MOV LS[ERROR.MASK] TO WR[1]                 ;current error mask
U 0A59, A0,C295   ;11806          COM WR[1]                                   ;flip it
U 0A5A, 37,3215   ;11807          MOV LS[PORT0] TO WR[0]                      ;received sector count bit
                  ;11808          AND WR[1] TO WR[0],                         ;mask it out to check for zero-state
U 0A5B, AF,C235   ;11809           DT(LONG)&SET.ALU.CC
U 0A5C, 08,A611   ;11810          JMP.IF[NEQ] TO [CHECK.TIMEOUT.0]            ;not zero, see if time loop expires
                  ;11811
                  ;11812 ;If check O.K., set the zero-or-one flag and initialize the check-for-one loop counter :
                  ;11813
U 0A5D, B6,4015   ;11814          MOV LS[#1] TO WR[0]
U 0A5E, BE,1415   ;11815          MOV WR[0] TO LS[T10]                        ;set the zero-or-one flag
U 0A5F, 65,1215   ;11816          CLR LS[T9]                                  ;init. the check-for-one loop counter
U 0A60, 08,A3E4   ;11817          JMP [T.A.3]                                 ;do another pass
                  ;11818
                  ;11819 ;Check if time expires waiting for sector count bit to be in zero-state:
                  ;11820
                  ;11821 CHECK.TIMEOUT.0:
U 0A61, B7,8215   ;11822          MOV LS[SECNT.TIME] TO WR[0]                 ;time allowed for wait
                  ;11823          CMP WR[0] WITH LS[T8],                      ;time expires ??
U 0A62, CF,1035   ;11824           DT(LONG)&SET.ALU.CC
U 0A63, 88,A6B9   ;11825          JMP.IF[EQL] TO [TA.ERR.1]                   ;yes, report error 1
U 0A64, 7F,1015   ;11826          INC LS[T8]                                  ;inc. check-for-zero loop counter
U 0A65, 08,A3E4   ;11827          JMP [T.A.3]                                 ;do another pass
                  ;11828
                  ;11829 ;Check if time expires waiting for sector count bit to be in one-state:
                  ;11830
                  ;11831 CHECK.TIMEOUT.1:
U 0A66, B7,8215   ;11832          MOV LS[SECNT.TIME] TO WR[0]                 ;time allowed for wait
                  ;11833          CMP WR[0] WITH LS[T9],                      ;time expires ??
U 0A67, 4F,1235   ;11834           DT(LONG)&SET.ALU.CC
U 0A68, 08,A749   ;11835          JMP.IF[EQL] TO [TA.ERR.2]                   ;yes, report error 2
U 0A69, FF,1215   ;11836          INC LS[T9]                                  ;inc. check-for-zero loop counter
U 0A6A, 08,A3E4   ;11837          JMP [T.A.3]                                 ;do another pass
                  ;11838
                  ;11839 ;Timeout on waiting to encounter the desired sector count bit in zero-state :
                  ;11840
                  ;11841 TA.ERR.1:
U 0A6B, B6,4015   ;11842          MOV LS[#1] TO WR[0]                         ;error number = 1
U 0A6C, BE,8215   ;11843          MOV WR[0] TO LS[ERROR.NUMBER]
U 0A6D, B6,4E95   ;11844          MOV LS[R80] TO WR[1]                        ;R80 as module code to indicate
U 0A6E, BE,8C95   ;11845          MOV WR[1] TO LS[MODULE.NUM]                 ;  module at fault
U 0A6F, 37,3215   ;11846          MOV LS[PORT0] TO WR[0]                      ;received data from port
U 0A70, 2F,8295   ;11847          CLR WR[1]                                   ;expected data
U 0A71, 08,693C   ;11848          JSR [CHECK.RESULT]                          ;we know this was not equal !!
U 0A72, 08,A6B4   ;11849          JMP [TA.ERR.1]                              ;err and LOE is set
U 0A73, 08,A3E4   ;11850          JMP [T.A.3]                                 ;continue
                  ;11851
                  ;11852 ;Timeout on waiting to encounter the desired sector count bit in one-state :
                  ;11853
                  ;11854 TA.ERR.2:
U 0A74, 36,4215   ;11855          MOV LS[#2] TO WR[0]                         ;error number = 2
U 0A75, BE,8215   ;11856          MOV WR[0] TO LS[ERROR.NUMBER]
U 0A76, B6,4E95   ;11857          MOV LS[R80] TO WR[1]                        ;R80 as module code to indicate
U 0A77, BE,8C95   ;11858          MOV WR[1] TO LS[MODULE.NUM]                 ;  module at fault
U 0A78, 37,3215   ;11859          MOV LS[PORT0] TO WR[0]                      ;received data from port
```

```
U 0A79, 37,2E95  ;11860          MOV LS[#1F] TO WR[1]                 ;expected data
U 0A7A, 08,693C  ;11861          JSR [CHECK.RESULT]                   ;we know this was not equal !!
U 0A7B, 88,A744  ;11862          JMP [TA.ERR.2]                       ;err and LOE is set
U 0A7C, 08,A3E4  ;11863          JMP [T.A.3]                          ;continue
                 ;11864
                 ;11865
                 ;11866 END.TA:                                       ;end of test
```

```
.PAGE  "Test B  R80 READ HEADER AND SEEK TEST (not run in CRD mode)"
;**
;
; Test Description:
;
;         This test will first issue the recalibrate command by setting DAR
;         <bit 15> high before issuing a seek command to bring the heads on
;         track 0 and then it will do a number of seeks to position the heads
;         in different locations in order to test the TAG BUS signals and the
;         validity of this function will be verified by read header function.
;         This test will first load the DAR with a specific cylinder address
;         and seek function will be issued by writing to the CSR <bit 2 and
;         1>. After CRDY is set, the read header function will be issued by
;         writing to the CSR <bit 3> and the first two bytes of FIFO A with
;         the cylinder address of the heads will be read and compared with
;         the cylinder address anticipated.
;
; Assumption:
;
;         Both CPU micros and IDC part I has been run successfully and
;         drive 0 is a R80, up and spinning and ready to accept commands;
;         R80 get status function is working and the R80 has the expected
;         operational status.
;
;         This test assumes "DISK DRIVE SIZING TEST" has been run prior to this
;         test.
;
; Test steps:
;
;           1. Load LS[DR0.R80] and examine its content
;           2. If value is zero, go to end of test
;           3. Else set DAR bits <6 & 15>
;           4. Write CSR with bits <2 & 1> set for recalibration
;           5. Set up a time out loop waiting for CRDY to set
;           6. If time expires for CRDY to set,
;           7.       Then set up pertinent error parameters for ERROR 1
;           8.             Force an error in JSR[CHECK.RESULT]
;           9.       Else write CSR with bits <3> set for read header
;          10. Set up a time out loop waiting for CRDY to set
;          11. If time expires for CRDY to set,
;          12.       Then set up pertinent error parameters for ERROR 2
;          13.             Force an error in JSR[CHECK.RESULT]
;          14.       Else check for bad sector, if sector bad;
;          15.              Then check if timeout on read another sector; if timeout
;          16.                   Then report ERROR 5
;          17.                   Else inc. time counter and go to step 9
;          18.              Else read the cylinder address from FIFO A
;          19. Compare received cylinder with zero by JSR[CHECK.RESULT]
;          20. If error, then report ERROR 3
;          21.             Else init. cyl. counter
;          22. Shift cylinder counter left by one bit
;          23. Set up DAR with the current cyl. counter
;          24. Write CSR with bits <2 & 1> set for seek
;          25. Wait for CRDY to set
;          26. Write CSR with bit <3> set for read header
```

```
;11922 ;         27. Wait for CRDY to set
;11923 ;         28. Check for bad sector, if sector bad;
;11924 ;         29.     Then check if timeout on read another sector; if timeout
;11925 ;         30.             Then report ERROR 6
;11926 ;         31.             Else inc. time counter and go to step 26
;11927 ;         32.     Else read the cylinder address from FIFO A
;11928 ;         33. Compare rec. with current cyl. by JSR[CHECK.RESULT]
;11929 ;         34. If error, then report ERROR 4
;11930 ;         35. Else repeat from step 22 to 33 nine times
;11931 ;         36. Test ends
;11932 ;
;11933 ; Errors:
;11934 ;
;11935 ; ERROR 1 - Time out waiting for CRDY to set after a Seek command has been
;11936 ;           issued to the IDC
;11937 ;
;11938 ; ERROR 2 - Time out waiting for CRDY to set after a Read header command has
;11939 ;           been issued to the IDC
;11940 ;
;11941 ;         Printout:
;11942 ;              EXPECTED        RECEIVED         MODULE
;11943 ;              N/A             N/A              IDC
;11944 ;
;11945 ; ERROR 3 & 4 - seek command did not bring the heads to the right
;11946 ;               cylinder address.
;11947 ;
;11948 ;         Printout:
;11949 ;              EXPECTED        RECEIVED         MODULE
;11950 ;              cyl. adr.       cyl. adr.        R80
;11951 ;
;11952 ; ERROR 5 & 6 - timeout on trying to find a good sector on the current cylinder
;11953 ;
;11954 ;         Printout:
;11955 ;              EXPECTED        RECEIVED         OTHER           MODULE
;11956 ;              cyl. adr.       cyl. adr.        CSR status      R80
;11957 ;
;11958 ; Debug:
;11959 ;
;11960 ; ERROR 1: Check if DRDY signal is at fault
;11961 ;
;11962 ; ERROR 2: Check R80 READ CLK from the R80 disk drive and the mux 74S158 and
;11963 ;          all signals going through it such as the R80 COMB SEC PLS, R80
;11964 ;          READ DATA, R80 SERVO CLK, R80 READ CLK, DS CLK and their output
;11965 ;          signals.
;11966 ;
;11967 ; ERROR 3 & 4: Check the R80 TAG BUS bits <9:0> and the R80 TAG coming out of
;11968 ;              the 3-state differential drivers 75113. It is possible that the
;11969 ;              the disk drive won't be able to get to its designated cylinders
;11970 ;              if these TAG lines are at fault.
;11971 ;
;11972 ; ERROR 5 & 6: Bad sectors are found on the disk at a particular cylinder
;11973 ;              and the disk address could be examined by looking into the
;11974 ;              CSR data under "OTHER" of the error report.
;11975 ;
;11976 ;--
```

```
                    ;11977
                    ;11978 T.B:
U 0A7D, B6,5E15     ;11979          MOV LS[BEGIN.TEST] TO WR[0]                 ;set bit 15 in WR[0] for
                    ;11980                                                      ; control and status word
U 0A7E, 3E,8015     ;11981          MOV WR[0] TO LS[CONTROL.STATUS]             ;set bit 15 in control and
                    ;11982                                                      ; status word - bit 15
                    ;11983                                                      ; indicates start of test to
                    ;11984                                                      ; console
U 0A7F, 10,E015     ;11985          MISC [SET.CP.ATTN]                          ;issue CPU ATTN to console
                    ;11986
                    ;11987 WAIT.TB.0:
U 0A80, 08,A804     ;11988          JMP [WAIT.TB.0]                             ;loop here waiting for
                    ;11989                                                      ; console to respond
                    ;11990
U 0A81, 65,A015     ;11991          CLR LS[PREVIOUS.ERROR]                      ;clear previous error number
                    ;11992                                                      ; in LS
U 0A82, 17,CC15     ;11993          PORT.CONTROL [CLEAR.IDC]                    ;init. IDC
U 0A83, 17,CC15     ;11994          PORT.CONTROL [CLEAR.IDC]
U 0A84, 17,CC15     ;11995          PORT.CONTROL [CLEAR.IDC]
                    ;11996
                    ;11997 T.B.1:
U 0A85, B7,0C15     ;11998          MOV LS[DR0.R80] TO WR[0]                    ;load the R80 presence indicator
                    ;11999          TST WR[0],                                  ;see if drive one has an R80
U 0A86, 20,0035     ;12000           DT(LONG)&SET.ALU.CC
U 0A87, 08,B389     ;12001          JMP.IF[EQL] TO [END.TB]                     ;no R80, go to end of test
                    ;12002
U 0A88, E5,1C15     ;12003          CLR LS[T14]                                 ;init. read header counter
                    ;12004
                    ;12005 ;Issue a recalibrate function through the seek command to bring the heads to
                    ;12006 ; cylinder zero :
                    ;12007
                    ;12008 T.B.2:
U 0A89, B6,4C15     ;12009          MOV LS[BIT6] TO WR[0]
U 0A8A, C7,5E15     ;12010          BIS LS[BIT15] TO WR[0]                      ;DAR bits for recalibrate
U 0A8B, 97,A415     ;12011          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]        ;write to the DAR
                    ;12012
U 0A8C, 37,0415     ;12013          MOV LS[SEEK] TO WR[0]                       ;fetch the seek command
U 0A8D, 47,9C15     ;12014          BIS LS[SETCSR.DS0] TO WR[0]                 ;select drive zero
U 0A8E, 17,A015     ;12015          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]        ;start the R80 seek
                    ;12016
                    ;12017 ;*** R80 SEEK IDC FUNCTION CODES : ***
                    ;12018
                    ;12019 ;-------------------------------------------------------------------------
                    ;12020 ;=1
                    ;12021 ;
                    ;12022 ;TEST.OR.RUN:
                    ;12023 ;
                    ;12024 ;        BEN2/MAINT,             ;IS THAT YOU, VICKIE?
                    ;12025 ;        NAD/DRIVE.TYPE
                    ;12026 ;
                    ;12027 ;-------------------------------------------------------------------------
                    ;12028 ;=0**
                    ;12029 ;
                    ;12030 ;DRIVE.TYPE:
                    ;12031 ;
```

```
;12032 ;        BENO/R80,                       ;WHICH TYPE OF DRIVE WAS SELECTED?
;12033 ;        NAD/RL02.IFORK
;12034 ;
;12035 ;--------------------------------------------------------------------------------
;12036 ;=1
;12037 ;
;12038 ;R80.IFORK:
;12039 ;
;12040 ;        UCMD/SET.FIFOA,
;12041 ;        FUNC/CLR.ECC.ERROR,             ; Enable clear of ECC/CRC errors for new function
;12042 ;        BENO/FO,                        ;8 WAY INSTRUCTION FORK.
;12043 ;        BEN1/F1,                        ;BRANCH ON F<2:0> IN THE CSR.
;12044 ;        BEN2/F2,
;12045 ;        NAD/#NOP.CMD
;12046 ;
;12047 ;--------------------------------------------------------------------------------
;12048 ;=011
;12049 ;
;12050 ;#SEEK.CMD:
;12051 ;
;12052 ;        NAD/#WAIT.DRDY3                 ;START OF THE R80 SEEK COMMAND
;12053 ;
;12054 ;--------------------------------------------------------------------------------
;12055 ;=0
;12056 ;
;12057 ;#WAIT.DRDY3:
;12058 ;
;12059 ;        BENO/DRIVE.RDY,                 ;WAIT FOR DRIVE READY
;12060 ;        NAD/#WAIT.DRDY3
;12061 ;
;12062 ;--------------------------------------------------------------------------------
;12063 ;=1
;12064 ;
;12065 ;#ENB.TAG.DATA3:
;12066 ;
;12067 ;        SEEK.INSTR/ON,                  ;DAR GATED TO R80 VIA TAG DATA LINES
;12068 ;        LOAD.LOOP.CNTR.[OF6]
;12069 ;
;12070 ;--------------------------------------------------------------------------------
;12071 ;=0
;12072 ;
;12073 ;#STB.TAG.STROBE3:
;12074 ;
;12075 ;        TAG.STROBE/ON,                  ;ASSERT CONTROL TAG STROBE
;12076 ;        SEEK.INSTR/ON,
;12077 ;        INCR.LPCNTR/ON,
;12078 ;        BENO/CNTR.OVFLW,
;12079 ;        NAD/#STB.TAG.STROBE3
;12080 ;
;12081 ;--------------------------------------------------------------------------------
;12082 ;=1
;12083 ;
;12084 ;#HOLD.TAG.DATA3:
;12085 ;
;12086 ;        SEEK.INSTR/ON                   ;KEEP DATA ASSERTED FOR 1 MORE USTATE
```

```
;12087 ;
;12088 ;------------------------------------------------------------------------
;12089 ;
;12090 ;#SEEK.CMD.DONE3:
;12091 ;
;12092 ;        FUNC/CLR.ONLINE,           ; Avoid setting attn bit on -DRDY
;12093 ;        NAD/#XFER.DONE4            ;RETURN TO START
;12094 ;
;12095 ;------------------------------------------------------------------------
;12096 ;
;12097 ;#XFER.DONE4:
;12098 ;
;12099 ;        UCMD/SET.CRDY,             ;SIGNAL END OF TRANSFER
;12100 ;        FUNC/SET.INT,
;12101 ;        NAD/SET.INT.AND.IDLE
;12102 ;
;12103 ;------------------------------------------------------------------------
;12104 ;=111111110
;12105 ;
;12106 ;SET.INT.AND.IDLE:
;12107 ;
;12108 ;        FUNC/SET.INT,              ; The purpose of this state is to give CRDY one
;12109 ;        NAD/IDLE                   ; state to reset before asserting interrupt.
;12110 ;                                   ; CRDY is now used in the INT equation.
;12111 ;
;12112 ;------------------------------------------------------------------------
;12113 ;=1
;12114 ;
;12115 ;RTN.TO.IDLE:
;12116 ;IDLE:
;12117 ;
;12118 ;        BENO/CRDY.L,               ;LOOK FOR GO (CRDY L)
;12119 ;        NAD/SETUP.DRV.CHK
;12120 ;
;12121 ;------------------------------------------------------------------------
;12122 ;=0
;12123 ;
;12124 ;SETUP.DRV.CHK:
;12125 ;
;12126 ;        LOAD.LOOP.CNTR.[OFB],      ;COUNTER WILL BE USED
;12127 ;        NAD/SEL.NXT.DRV            ;TO SEQUENCE THRU DRIVES.
;12128 ;
;12129 ;------------------------------------------------------------------------
;12130 ;
;12131 ;SEL.NXT.DRV:
;12132 ;
;12133 ;        SEL.DRV/UPROG,
;12134 ;        BENO/CNTR.OVFLW,
;12135 ;        NAD/CHK.DRV.STATUS
;12136 ;
;12137 ;------------------------------------------------------------------------
;12138 ;=0
;12139 ;
;12140 ;CHK.DRV.STATUS:
;12141 ;        ; Give drive status time to settle down.
```

```
                      ;12142 ;
                      ;12143 ;         SEL.DRV/UPROG,            ;SELECT DRIVE
                      ;12144 ;         NAD/DRV.STATUS.DISPATCH
                      ;12145 ;
                      ;12146 ;-----------------------------------------------------------------
                      ;12147 ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
                      ;12148 ;
                      ;12149 ;DRV.STATUS.DISPATCH:
                      ;12150 ;         SEL.DRV/UPROG,            ;SELECT DRIVE
                      ;12151 ;         BEN1/DRIVE.ONLINE,        ;AND BRANCH ON STATUS.
                      ;12152 ;         BEN0/DRIVE.RDY,
                      ;12153 ;         NAD/-ERR.-OL.-RDY
                      ;12154 ;
                      ;12155 ;
                      ;12156 ;-----------------------------------------------------------------
                      ;12157 ;=011
                      ;12158 ;
                      ;12159 ;-ERR.OL.RDY:
                      ;12160 ;
                      ;12161 ;         SEL.DRV/UPROG,            ;NOP
                      ;12162 ;         INCR.LPCNTR/ON,           ;ADD 1 TO DRIVE SELECT.
                      ;12163 ;         NAD/SEL.NXT.DRV
                      ;12164 ;
                      ;12165 ;-----------------------------------------------------------------
                      ;12166 ;
                      ;12167 ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                      ;12168 ;
                      ;12169 ;                UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                      ;12170 ;
                      ;12171 ;-----------------------------------------------------------------
                      ;12172 ;
                      ;12173 ;=1
                      ;12174 ;
                      ;12175 ;RTN.TO.IDLE:
                      ;12176 ;IDLE:
                      ;12177 ;
                      ;12178 ;         BEN0/CRDY.L,              ;LOOK FOR GO (CRDY L)
                      ;12179 ;         NAD/SETUP.DRV.CHK
                      ;12180 ;
                      ;12181 ;-----------------------------------------------------------------
                      ;12182
                      ;12183 ;Set up parameters for timeout error report :
                      ;12184
U 0A8F, 36,4095       ;12185           MOV LS[#1] TO WR[1]
U 0A90, 3E,8295       ;12186           MOV WR[1] TO LS[ERROR.NUMBER]           ;set up error number = 1 in LS
U 0A91, E5,8A15       ;12187           CLR LS[ERROR.MASK]                      ;no error mask used here
U 0A92, 36,4A95       ;12188           MOV LS[IDC] TO WR[1]                    ;store module code in LS to
U 0A93, BE,8C95       ;12189           MOV WR[1] TO LS[MODULE.NUM]             ; modules
U 0A94, 36,6E95       ;12190           MOV LS[NA.EXP.REC] TO WR[1]             ;put "N/A" for printout data
U 0A95, BE,8095       ;12191           MOV WR[1] TO LS[CONTROL.STATUS]
                      ;12192
U 0A96, 2F,8015       ;12193           CLR WR[0]                               ;Initialize time-out counter
U 0A97, BF,8C15       ;12194           MOV WR[0] TO LS[TIMEOUT.CNTR]
                      ;12195
                      ;12196 WAIT.IDC.TB.1:
```

```
U 0A98, 37,8C15   ;12197          MOV LS[TIMEOUT.CNTR] TO WR[0]                ;load the current time count
U 0A99, 37,9095   ;12198          MOV LS[WAIT.TIME1] TO WR[1]                  ;time limit on seek
                  ;12199          CMP WR[0] WITH WR[1],                        ;timeout yet ???
U 0A9A, 26,40B5   ;12200           DT(LONG)&SET.ALU.CC
U 0A9B, 88,AA19   ;12201          JMP.IF[EQL] TO [TB.ERR.1]                    ;yes, timeout on seek
U 0A9C, 20,4015   ;12202          INC WR[0]                                    ;inc. time count
U 0A9D, BF,8C15   ;12203          MOV WR[0] TO LS[TIMEOUT.CNTR]
U 0A9E, 88,700C   ;12204          JSR [WAIT.CSR.CRDY]                          ;check if CRDY is set
U 0A9F, 88,AA64   ;12205          JMP [CHECK.DRDY.TB.1]                        ;yes, CRDY is set now
U 0AA0, 08,A984   ;12206          JMP [WAIT.IDC.TB.1]                          ;no, wait some more
                  ;12207
                  ;12208 ;Force an error in the CHECK.RESULT routine for this timeout :
                  ;12209
                  ;12210 TB.ERR.1:
U 0AA1, 2F,8015   ;12211          CLR WR[0]                                    ;set unequal values
U 0AA2, 36,4095   ;12212          MOV LS[#1] TO WR[1]                          ; between WR0 and WR1
U 0AA3, 08,693C   ;12213          JSR [CHECK.RESULT]                           ;force an error
U 0AA4, 08,A894   ;12214          JMP [T.B.2]                                  ;if LOE is set
U 0AA5, 88,AA64   ;12215          JMP [CHECK.DRDY.TB.1]                        ;check if R80 is ready
                  ;12216
                  ;12217 ;Check if the R80 is ready to take the next operation :
                  ;12218
                  ;12219 CHECK.DRDY.TB.1:
U 0AA6, 08,708C   ;12220          JSR [WAIT.CSR.DRDY]                          ;check for DRDY bit set
                  ;12221
                  ;12222 ;Do a read header command to ensure the heads are on cylinder zero :
                  ;12223
                  ;12224 T.B.3:
U 0AA7, B7,0615   ;12225          MOV LS[READ.HEADER] TO WR[0]                 ;fetch the read header command
U 0AA8, 47,9C15   ;12226          BIS LS[SETCSR.DS0] TO WR[0]                  ;select drive zero
U 0AA9, 17,A015   ;12227          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]         ;issue the read header command
                  ;12228
                  ;12229 ;*** R80 READ HEADER IDC FUNCTION CODES : ***
                  ;12230
                  ;12231 ;----------------------------------------------------------------------
                  ;12232 ;=1
                  ;12233 ;
                  ;12234 ;TEST.OR.RUN:
                  ;12235 ;
                  ;12236 ;        BEN2/MAINT,                ;IS THAT YOU, VICKIE?
                  ;12237 ;        NAD/DRIVE.TYPE
                  ;12238 ;
                  ;12239 ;----------------------------------------------------------------------
                  ;12240 ;=0**
                  ;12241 ;
                  ;12242 ;DRIVE.TYPE:
                  ;12243 ;
                  ;12244 ;        BEN0/R80,                  ;WHICH TYPE OF DRIVE WAS SELECTED?
                  ;12245 ;        NAD/RL02.IFORK
                  ;12246 ;
                  ;12247 ;----------------------------------------------------------------------
                  ;12248 ;=1
                  ;12249 ;
                  ;12250 ;R80.IFORK:
                  ;12251 ;
```

```
;       UCMD/SET.FIFOA,
;       FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
;       BEN0/F0,                ;8 WAY INSTRUCTION FORK.
;       BEN1/F1,                ;BRANCH ON F<2:0> IN THE CSR.
;       BEN2/F2,
;       NAD/@NOP.CMD
;
;-----------------------------------------------------------------------
;=100
;
;@RD.HDR.CMD:
;
;       NAD/@WAIT.DRDY4         ;START OF THE R80 READ HEADER COMMAND
;
;-----------------------------------------------------------------------
;=0
;
;@WAIT.DRDY4:
;
;       BEN0/DRIVE.RDY,         ;WAIT FOR DRIVE READY
;       NAD/@WAIT.DRDY4
;
;-----------------------------------------------------------------------
;=1
;
;@CHGCLK4:
;
;       ENB.SYS.CLK,            ;USE R80 SYSTEM CLOCK
;       NAD/@WAIT.NOSEC.PLS4
;
;-----------------------------------------------------------------------
;=1*
;
;@WAIT.NOSEC.PLS4:
;
;       BEN1/SECTOR.PLS,        ;PREPARE TO LOOK FOR THE LEADING EDGE
;       NAD/@WAIT.NOSEC.PLS4    ;OF SECTOR PULSE.
;
;-----------------------------------------------------------------------
;=0*
;
;@NOSEC.PLS4:
;
;       NAD/@WAIT.SEC.PLS4      ;WE ARE NOT IN SECTOR PULSE AREA.
;
;-----------------------------------------------------------------------
;=0*
;
;@WAIT.SEC.PLS4:
;
;       BEN1/SECTOR.PLS,        ;WAIT FOR LEADING EDGE OF SEC PLS
;       NAD/@WAIT.SEC.PLS4
;
;-----------------------------------------------------------------------
;=1*
```

```
;12307 ;
;12308 ;#SEC.PLS.FND4:
;12309 ;
;12310 ;       CLR.FIFO.ADR,             ;SECTOR PULSE HAS BEEN FOUND
;12311 ;       LOAD.LOOP.CNTR.[0C5]
;12312 ;
;12313 ;----------------------------------------------------------------
;12314 ;=0
;12315 ;
;12316 ;#WAIT.60.TICKS4:
;12317 ;
;12318 ;       INCR.LPCNTR/ON,           ;DELAY BEFORE ASSERTING READ GATE
;12319 ;       TAG.STROBE/ON,
;12320 ;       BENO/CNTR.OVFLW,
;12321 ;       NAD/#WAIT.60.TICKS4
;12322 ;
;12323 ;----------------------------------------------------------------
;12324 ;=1
;12325 ;
;12326 ;#SETUP.LOOP.LOCK4:
;12327 ;
;12328 ;       LOOP.LOCK/ON,             ;ALLOW PHASE LOCK LOOP TO SYNC ON
;12329 ;       TAG.STROBE/ON,            ;ALL ZEROS PATTERN
;12330 ;       LOAD.LOOP.CNTR.[0A9]
;12331 ;
;12332 ;----------------------------------------------------------------
;12333 ;=0
;12334 ;
;12335 ;#WAIT.LOOP.LOCK4:
;12336 ;
;12337 ;       LOOP.LOCK/ON,             ;WAIT FOR 88 TICKS AFTER READ GATE
;12338 ;       TAG.STROBE/ON,
;12339 ;       INCR.LPCNTR/ON,
;12340 ;       BENO/CNTR.OVFLW,
;12341 ;       NAD/#WAIT.LOOP.LOCK4
;12342 ;
;12343 ;----------------------------------------------------------------
;12344 ;=1
;12345 ;
;12346 ;#SETUP.RD.HDR4:
;12347 ;
;12348 ;       LOOP.LOCK/ON,             ;SETUP INITIAL CONDITIONS
;12349 ;       TAG.STROBE/ON,            ;FOR READING HEADER.
;12350 ;       FUNC/CLR.DSR,
;12351 ;       LOAD.LOOP.CNTR.[0F9]
;12352 ;
;12353 ;----------------------------------------------------------------
;12354 ;
;12355 ;#CHGCLK4A:
;12356 ;
;12357 ;       ENB.DSK.CLK,              ;THE MICROSEQUENCER WILL STALL
;12358 ;       LOOP.LOCK/ON,             ;UNTIL SYNC SEEN IS SET
;12359 ;       TAG.STROBE/ON,            ;(SYNC BYTE FOUND)
;12360 ;       GEN.SYNC.[19]             ;THEN WE USE READ CLK FROM R80.
;12361 ;
```

```
;12362 ;-------------------------------------------------------------------------
;12363 ;=0
;12364 ;
;12365 ;@READ.HDR.BYTE4:
;12366 ;
;12367 ;        LOOP.LOCK/ON,               ;READ THE FIRST HEADER BYTE
;12368 ;        TAG.STROBE/ON,
;12369 ;        INCR.LPCNTR/ON,
;12370 ;        BENO/CNTR.OVFLW,
;12371 ;        NAD/@READ.HDR.BYTE4
;12372 ;
;12373 ;-------------------------------------------------------------------------
;12374 ;=1
;12375 ;
;12376 ;@STORE.HDR.BYTE4:
;12377 ;
;12378 ;        LOOP.LOCK/ON,               ;STORE FIRST HDR BYTE IN FIFO
;12379 ;        TAG.STROBE/ON,
;12380 ;        ENB.DSR/ON,
;12381 ;        WRT.FIFO/ON,
;12382 ;        INCR.FIFO.CNTR/ON,
;12383 ;        LOAD.LOOP.CNTR.[0F9]
;12384 ;
;12385 ;-------------------------------------------------------------------------
;12386 ;=0
;12387 ;
;12388 ;@READ.HDR.BYTE4A:
;12389 ;
;12390 ;        LOOP.LOCK/ON,               ;READ SECOND HDR BYTE
;12391 ;        TAG.STROBE/ON,
;12392 ;        INCR.LPCNTR/ON,
;12393 ;        BENO/CNTR.OVFLW,
;12394 ;        NAD/@READ.HDR.BYTE4A
;12395 ;
;12396 ;-------------------------------------------------------------------------
;12397 ;=1
;12398 ;
;12399 ;@STORE.HDR.BYTE4A:
;12400 ;
;12401 ;        LOOP.LOCK/ON,               ;STORE SECOND HDR BYTE IN FIFO
;12402 ;        TAG.STROBE/ON,
;12403 ;        ENB.DSR/ON,
;12404 ;        WRT.FIFO/ON,
;12405 ;        INCR.FIFO.CNTR/ON,
;12406 ;        LOAD.LOOP.CNTR.[0F9]
;12407 ;
;12408 ;-------------------------------------------------------------------------
;12409 ;=0
;12410 ;
;12411 ;@READ.HDR.BYTE4B:
;12412 ;
;12413 ;        LOOP.LOCK/ON,               ;READ THE THIRD HEADER BYTE
;12414 ;        TAG.STROBE/ON,
;12415 ;        INCR.LPCNTR/ON,
;12416 ;        BENO/CNTR.OVFLW,
;12417 ;        NAD/#READ.HDR.BYTE4B
;12418 ;
;12419 ;-------------------------------------------------------------------
;12420 ;=1
;12421 ;
;12422 ;#STORE.HDR.BYTE4B:
;12423 ;
;12424 ;        LOOP.LOCK/ON,              ;STORE THIRD HDR BYTE IN FIFO
;12425 ;        TAG.STROBE/ON,
;12426 ;        ENB.DSR/ON,
;12427 ;        WRT.FIFO/ON,
;12428 ;        INCR.FIFO.CNTR/ON,
;12429 ;        LOAD.LOOP.CNTR.[OF9]
;12430 ;
;12431 ;-------------------------------------------------------------------
;12432 ;=0
;12433 ;
;12434 ;#READ.HDR.BYTE4C:
;12435 ;
;12436 ;        LOOP.LOCK/ON,              ;READ FOURTH HDR BYTE
;12437 ;        TAG.STROBE/ON,
;12438 ;        INCR.LPCNTR/ON,
;12439 ;        BENO/CNTR.OVFLW,
;12440 ;        NAD/#READ.HDR.BYTE4C
;12441 ;
;12442 ;-------------------------------------------------------------------
;12443 ;=1
;12444 ;
;12445 ;#STORE.HDR.BYTE4C:
;12446 ;
;12447 ;        LOOP.LOCK/ON,              ;STORE FOURTH HDR BYTE IN FIFO
;12448 ;        TAG.STROBE/ON,
;12449 ;        ENB.DSR/ON,
;12450 ;        WRT.FIFO/ON,
;12451 ;        INCR.FIFO.CNTR/ON,
;12452 ;        LOAD.LOOP.CNTR.[OF9]
;12453 ;
;12454 ;-------------------------------------------------------------------
;12455 ;=0
;12456 ;
;12457 ;#READ.CRC.BYTE4:
;12458 ;
;12459 ;        LOOP.LOCK/ON,              ;READ FIRST HEADER CRC BYTE
;12460 ;        TAG.STROBE/ON,
;12461 ;        INCR.LPCNTR/ON,
;12462 ;        BENO/CNTR.OVFLW,
;12463 ;        NAD/#READ.CRC.BYTE4
;12464 ;
;12465 ;-------------------------------------------------------------------
;12466 ;=1
;12467 ;
;12468 ;#STORE.CRC.BYTE4:
;12469 ;
;12470 ;        LOOP.LOCK/ON,              ;STORE FIRST HDR CRC BYTE IN FIFO
;12471 ;        TAG.STROBE/ON,
;12472 ;         ENB.DSR/ON,
;12473 ;         WRT.FIFO/ON,
;12474 ;         INCR.FIFO.CNTR/ON,
;12475 ;         LOAD.LOOP.CNTR.[0F9]
;12476 ;
;12477 ;------------------------------------------------------------------------
;12478 ;=0
;12479 ;
;12480 ;@READ.CRC.BYTE4A:
;12481 ;
;12482 ;         LOOP.LOCK/ON,              ;READ SECOND HDR CRC BYTE
;12483 ;         TAG.STROBE/ON,
;12484 ;         INCR.LPCNTR/ON,
;12485 ;         BENO/CNTR.OVFLW,
;12486 ;         NAD/@READ.CRC.BYTE4A
;12487 ;
;12488 ;------------------------------------------------------------------------
;12489 ;=1
;12490 ;
;12491 ;@STORE.CRC.BYTE4A:
;12492 ;
;12493 ;         LOOP.LOCK/ON,              ;STORE SECOND HDR CRC BYTE IN FIFO
;12494 ;         TAG.STROBE/ON,
;12495 ;         ENB.DSR/ON,
;12496 ;         INCR.FIFO.CNTR/ON,
;12497 ;         WRT.FIFO/ON
;12498 ;
;12499 ;------------------------------------------------------------------------
;12500 ;
;12501 ;@CLEAR.FIFO.ADRS4:
;12502 ;
;12503 ;         LOOP.LOCK/ON,
;12504 ;         TAG.STROBE/ON,
;12505 ;         CLR.FIFO.ADR,              ;CLEAR THE FIFO ADRS COUNTER.
;12506 ;         NAD/@CHGCLK4B
;12507 ;
;12508 ;------------------------------------------------------------------------
;12509 ;
;12510 ;@CHGCLK4B:
;12511 ;
;12512 ;         LOOP.LOCK/ON,              ;SWITCH OVER TO CPU CLOCK
;12513 ;         TAG.STROBE/ON,             ;DURING POSTAMBLE.
;12514 ;         ENB.CPU.CLK
;12515 ;
;12516 ;------------------------------------------------------------------------
;12517 ;
;12518 ;@XFER.DONE4:
;12519 ;
;12520 ;         UCMD/SET.CRDY,             ;SIGNAL END OF TRANSFER
;12521 ;         FUNC/SET.INT,
;12522 ;         NAD/SET.INT.AND.IDLE
;12523 ;
;12524 ;------------------------------------------------------------------------
;12525 ;=111111110
;12526 ;
```

```
;12527 ;SET.INT.AND.IDLE:
;12528 ;
;12529 ;        FUNC/SET.INT,              ; The purpose of this state is to give CRDY one
;12530 ;        NAD/IDLE                   ; state to reset before asserting interrupt.
;12531 ;                                   ; CRDY is now used in the INT equation.
;12532 ;
;12533 ;-----------------------------------------------------------------------------
;12534 ;=1
;12535 ;
;12536 ;RTN.TO.IDLE:
;12537 ;IDLE:
;12538 ;
;12539 ;        BENO/CRDY.L,               ;LOOK FOR GO (CRDY L)
;12540 ;        NAD/SETUP.DRV.CHK
;12541 ;
;12542 ;-----------------------------------------------------------------------------
;12543 ;=0
;12544 ;
;12545 ;SETUP.DRV.CHK:
;12546 ;
;12547 ;        LOAD.LOOP.CNTR.[OFB],      ;COUNTER WILL BE USED
;12548 ;        NAD/SEL.NXT.DRV            ;TO SEQUENCE THRU DRIVES.
;12549 ;
;12550 ;-----------------------------------------------------------------------------
;12551 ;
;12552 ;SEL.NXT.DRV:
;12553 ;
;12554 ;        SEL.DRV/UPROG,
;12555 ;        BENO/CNTR.OVFLW,
;12556 ;        NAD/CHK.DRV.STATUS
;12557 ;
;12558 ;-----------------------------------------------------------------------------
;12559 ;=0
;12560 ;
;12561 ;CHK.DRV.STATUS:
;12562 ;        ; Give drive status time to settle down.
;12563 ;
;12564 ;        SEL.DRV/UPROG,             ;SELECT DRIVE
;12565 ;        NAD/DRV.STATUS.DISPATCH
;12566 ;
;12567 ;-----------------------------------------------------------------------------
;12568 ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
;12569 ;
;12570 ;DRV.STATUS.DISPATCH:
;12571 ;        SEL.DRV/UPROG,             ;SELECT DRIVE
;12572 ;        BEN1/DRIVE.ONLINE,         ;AND BRANCH ON STATUS.
;12573 ;        BENO/DRIVE.RDY,
;12574 ;        NAD/-ERR.-OL.-RDY
;12575 ;
;12576 ;
;12577 ;-----------------------------------------------------------------------------
;12578 ;=011
;12579 ;
;12580 ;-ERR.OL.RDY:
;12581 ;
```

```
                  ;12582 ;        SEL.DRV/UPROG,          ;NOP
                  ;12583 ;        INCR.LPCNTR/ON,         ;ADD 1 TO DRIVE SELECT.
                  ;12584 ;        NAD/SEL.NXT.DRV
                  ;12585 ;
                  ;12586 ;----------------------------------------------------------------------
                  ;12587 ;
                  ;12588 ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                  ;12589 ;
                  ;12590 ;               UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                  ;12591 ;
                  ;12592 ;----------------------------------------------------------------------
                  ;12593 ;
                  ;12594 ;=1
                  ;12595 ;
                  ;12596 ;RTN.TO.IDLE:
                  ;12597 ;IDLE:
                  ;12598 ;
                  ;12599 ;        BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
                  ;12600 ;        NAD/SETUP.DRV.CHK
                  ;12601 ;
                  ;12602 ;----------------------------------------------------------------------
                  ;12603
                  ;12604 ;Set up parameters for time-out error report :
                  ;12605
U 0AAA, B6,4295   ;12606          MOV LS[#2] TO WR[1]
U 0AAB, 3E,8295   ;12607          MOV WR[1] TO LS[ERROR.NUMBER]           ;set up error number = 2 in LS
U 0AAC, E5,8A15   ;12608          CLR LS[ERROR.MASK]                      ;no error mask used here
U 0AAD, 36,4A95   ;12609          MOV LS[IDC] TO WR[1]                    ;store module code in LS to
U 0AAE, BE,8C95   ;12610          MOV WR[1] TO LS[MODULE.NUM]             ; modules
U 0AAF, 36,6E95   ;12611          MOV LS[NA.EXP.REC] TO WR[1]             ;put "N/A" for printout data
U 0AB0, BE,8095   ;12612          MOV WR[1] TO LS[CONTROL.STATUS]
                  ;12613
U 0AB1, 2F,8015   ;12614          CLR WR[0]                               ;Initialize time-out counter
U 0AB2, BF,8C15   ;12615          MOV WR[0] TO LS[TIMEOUT.CNTR]
                  ;12616
                  ;12617 WAIT.IDC.TB.2:
U 0AB3, 37,8C15   ;12618          MOV LS[TIMEOUT.CNTR] TO WR[0]           ;load the current time count
U 0AB4, 37,9095   ;12619          MOV LS[WAIT.TIME1] TO WR[1]             ;time limit on read header
                  ;12620          CMP WR[0] WITH WR[1],                   ;timeout yet ???
U 0AB5, 26,40B5   ;12621           DT(LONG)&SET.ALU.CC
U 0AB6, 88,ACB9   ;12622          JMP.IF[EQL] TO [TB.ERR.2]               ;yes, timeout on read header
U 0AB7, 20,4015   ;12623          INC WR[0]                               ;inc. time count
U 0AB8, BF,8C15   ;12624          MOV WR[0] TO LS[TIMEOUT.CNTR]
U 0AB9, 88,700C   ;12625          JSR [WAIT.CSR.CRDY]                     ;check if CRDY is set
U 0ABA, 5B,001E   ;12626          SKIP.IF[SKIP]                           ;yes, CRDY is set now
U 0ABB, 08,AB34   ;12627          JMP [WAIT.IDC.TB.2]                     ;no, wait some more
                  ;12628
                  ;12629 CHECK.BAD.SECTOR.TB.1:
U 0ABC, 17,D815   ;12630          PORT.CONTROL [SEL.FIFO.A]               ;read FIFO A
U 0ABD, 17,C015   ;12631          PORT.CONTROL [CLEAR.FIFO.CNTR]          ;clear FIFO A address counter
U 0ABE, 90,9015   ;12632          MISC [SET.PORT.I.O]                     ;select IDC
U 0ABF, 97,8C15   ;12633          PORT.READ PORT.ADRS[DATA.WORD]          ;read the header from FIFO A
U 0AC0, 3B,3215   ;12634          MOV PORT TO LS[PORT0]                   ; and load that in
U 0AC1, 37,3215   ;12635          MOV LS[PORT0] TO WR[0]                  ;  WR0 as received data
U 0AC2, B7,C295   ;12636          MOV LS[#C000] TO WR[1]                  ;mask for UF & MF bits of CSR
```

```
U 0AC3, 2F,C215   ;12637          AND WR[1] TO WR[0]                         ;look at UF & MF bits
                  ;12638          CMP WR[0] WITH WR[1],                      ;good sector ??
U 0AC4, 26,40B5   ;12639           DT(LONG)&SET.ALU.CC
U 0AC5, 88,AD09   ;12640          JMP.IF[EQL] TO [IDC.DONE.TB.2]             ;yes, good sector, go on
U 0AC6, 36,1D15   ;12641          MOV LS[T14] TO WR[2]                       ;bring in the time counter
                  ;12642          BIT LS[BIT5] WITH WR[2],                   ;time expired ??
U 0AC7, D9,4B35   ;12643           DT(LONG)&SET.ALU.CC
U 0AC8, 08,B1D1   ;12644          JMP.IF[BIT.SET] TO [BAD.SECTOR.TB.1]       ;yes, do an error report
U 0AC9, 7F,1C15   ;12645          INC LS[T14]                                ;not yet, inc. time counter
U 0ACA, 08,AA74   ;12646          JMP [T.B.3]                                ;do another read header
                  ;12647
                  ;12648 ;Force an error in the CHECK.RESULT routine for this timeout :
                  ;12649
                  ;12650 TB.ERR.2:
U 0ACB, 2F,8015   ;12651          CLR WR[0]                                  ;set unequal values
U 0ACC, 36,4095   ;12652          MOV LS[#1] TO WR[1]                        ; for WR0 and WR1
U 0ACD, 08,693C   ;12653          JSR [CHECK.RESULT]                         ;force an error
U 0ACE, 08,AA74   ;12654          JMP [T.B.3]                                ;if LOE is set
U 0ACF, 08,AD04   ;12655          JMP [IDC.DONE.TB.2]
                  ;12656
                  ;12657 ;Set up error parameters for ERROR 3 report :
                  ;12658
                  ;12659 IDC.DONE.TB.2:
U 0AD0, 36,C015   ;12660          MOV LS[#3] TO WR[0]                        ;error number = 3
U 0AD1, BE,8215   ;12661          MOV WR[0] TO LS[ERROR.NUMBER]
U 0AD2, B7,B415   ;12662          MOV LS[#FFFFFC00] TO WR[0]                 ;R80 cylinder error mask
U 0AD3, 3E,8A15   ;12663          MOV WR[0] TO LS[ERROR.MASK]                ; bits <9-0>, 1st byte
U 0AD4, B6,4E95   ;12664          MOV LS[R80] TO WR[1]                       ;R80 as module code to indicate
U 0AD5, BE,8C95   ;12665          MOV WR[1] TO LS[MODULE.NUM]                ;  module at fault
                  ;12666
                  ;12667 ;Read the R80 header from FIFO A to verfy if the heads are really at cylinder zero
                  ;12668
                  ;12669 T.B.4:
U 0AD6, 37,3215   ;12670          MOV LS[PORT0] TO WR[0]                     ;fetch the header
U 0AD7, B6,9C95   ;12671          MOV LS[#0] TO WR[1]                        ;expected cylinder = 0
U 0AD8, 08,693C   ;12672          JSR [CHECK.RESULT]                         ;heads at zero cylinder ??
U 0AD9, 08,AA74   ;12673          JMP [T.B.3]                                ;no, err and if LOE is set
U 0ADA, 88,ADB4   ;12674          JMP [INIT.TB.1]                            ;yes, next test
                  ;12675
                  ;12676 ;Seek to 9 different cylinders to verify the 9 tag bus bits are good :
                  ;12677 ;First initialize the cylinder counter :
                  ;12678
                  ;12679 INIT.TB.1:
U 0ADB, B6,4015   ;12680          MOV LS[BIT0] TO WR[0]                      ;start cylinder
U 0ADC, 3F,8815   ;12681          MOV WR[0] TO LS[CYL.CNTR]                  ; counter with 1
                  ;12682
                  ;12683 ;In order to do a seek on the R80, a head select operation must be first initiated
                  ;12684 ; and it is then followed by the actual seek :
                  ;12685
                  ;12686 HEAD.SELECT.1:
U 0ADD, 36,4215   ;12687          MOV LS[#2] TO WR[0]                        ;selecting the third head
U 0ADE, 47,5C15   ;12688          BIS LS[BIT14] TO WR[0]                     ;set head select code
U 0ADF, 97,A415   ;12689          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]       ;load IDC DAR
                  ;12690
U 0AE0, 37,0415   ;12691          MOV LS[SEEK] TO WR[0]                      ;fetch the seek command
```

```
U 0AE1, 47,9C15   ;12692          BIS LS[SETCSR.DS0] TO WR[0]               ;select drive zero
U 0AE2, 17,A015   ;12693          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;initiate the head select command
                  ;12694
                  ;12695 ;It is necessary to wait for CRDY before issuing another command or destroying the
                  ;12696 ; DAR :
                  ;12697
                  ;12698 WAIT.HEAD.SELECT.DONE:
U 0AE3, 88,700C   ;12699          JSR [WAIT.CSR.CRDY]                       ;check if CRDY is ready
U 0AE4, 08,AE64   ;12700          JMP [T.B.5]                               ;yes, CRDY is set now
U 0AE5, 08,AE34   ;12701          JMP [WAIT.HEAD.SELECT.DONE]               ;no, wait some more
                  ;12702
                  ;12703 ;This section of code is to insure that the proper attention bit is cleared after
                  ;12704 ; the implied head select :
                  ;12705
                  ;12706 T.B.5:
U 0AE6, 65,1E15   ;12707          CLR LS[T15]                               ;init. read header counter
U 0AE7, 36,0A15   ;12708          MOV LS[T5.SUB] TO WR[0]                   ;bring in the CSR word
U 0AE8, B6,6695   ;12709          MOV LS[BIT19] TO WR[1]                    ;CSR ATN3 bit
U 0AE9, 47,6495   ;12710          BIS LS[BIT18] TO WR[1]                    ;add CSR ATN2 bit
U 0AEA, AF,4215   ;12711          BIC WR[1] TO WR[0]                        ;reset ATN3:ATN2 bits
U 0AEB, 17,A015   ;12712          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;clear attn. bits in CSR
                  ;12713
                  ;12714 ;Start the actual seek to the desired cylinders :
                  ;12715
                  ;12716 ACTUAL.R80.SEEK:
U 0AEC, B7,8815   ;12717          MOV LS[CYL.CNTR] TO WR[0]                 ;cylinder address for DAR format
U 0AED, 47,5A15   ;12718          BIS LS[BIT13] TO WR[0]                    ;specify this is a cyl. adr. word
U 0AEE, 97,C415   ;12719          PORT.CONTROL [RESET.BR]                   ;clear interrupt caused by head select
U 0AEF, 97,A415   ;12720          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]      ;write to the DAR
                  ;12721
U 0AF0, 37,0415   ;12722          MOV LS[SEEK] TO WR[0]                     ;fetch the seek command
U 0AF1, 47,9C15   ;12723          BIS LS[SETCSR.DS0] TO WR[0]               ;select drive zero
U 0AF2, 17,A015   ;12724          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;start the R80 seek
                  ;12725
                  ;12726       ;**********************************************************************
                  ;12727       ;* REFER TO THE R80 SEEK IDC FUNCTION CODES AS LISTED IN THIS TEST ABOVE *
                  ;12728       ;**********************************************************************
                  ;12729
                  ;12730 WAIT.IDC.TB.3:
U 0AF3, 88,700C   ;12731          JSR [WAIT.CSR.CRDY]                       ;check if CRDY is set
U 0AF4, 58,001E   ;12732          SKIP.IF[SKIP]                             ;yes, next test
U 0AF5, 88,AF34   ;12733          JMP [WAIT.IDC.TB.3]                       ;wait some more
                  ;12734
                  ;12735 ;Set up error parameters for ERROR 4 in case heads are not at the right cylinder :
                  ;12736
                  ;12737 IDC.DONE.TB.3:
U 0AF6, 36,4415   ;12738          MOV LS[#4] TO WR[0]                       ;error number = 4
U 0AF7, BE,8215   ;12739          MOV WR[0] TO LS[ERROR.NUMBER]
U 0AF8, B7,B415   ;12740          MOV LS[#FFFFFC00] TO WR[0]                ;R80 cylinder error mask
U 0AF9, 3E,8A15   ;12741          MOV WR[0] TO LS[ERROR.MASK]               ; bits <9-0>, 1st byte
U 0AFA, B6,4E95   ;12742          MOV LS[R80] TO WR[1]                      ;R80 as module code to indicate
U 0AFB, BE,8C95   ;12743          MOV WR[1] TO LS[MODULE.NUM]               ;  module at fault
U 0AFC, E5,8015   ;12744          CLR LS[CONTROL.STATUS]                    ;clr LS[NA.EXP.REC]
                  ;12745
                  ;12746 ;Do a read header to the R80 to find out the disk address the heads are at :
```

```
                ;12747
                ;12748 T.B.6:
U 0AFD, B7,0615 ;12749         MOV LS[READ.HEADER] TO WR[0]             ;fetch the read header command
U 0AFE, 47,9C15 ;12750         BIS LS[SETCSR.DS0] TO WR[0]              ;select drive zero
U 0AFF, 17,A015 ;12751         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]     ;issue the read header command
                ;12752
                ;12753         ;*** Refer to the R80 read header IDC function codes listed above ***
                ;12754
                ;12755 WAIT.IDC.TB.4:
U 0B00, 88,700C ;12756         JSR [WAIT.CSR.CRDY]                      ;check if CRDY is set
U 0B01, 5B,001E ;12757         SKIP.IF[SKIP]                            ;yes, it is set
U 0B02, 08,B004 ;12758         JMP [WAIT.IDC.TB.4]                      ;wait some more
                ;12759
                ;12760 ;Check for bad sector :
                ;12761
                ;12762 CHECK.BAD.SECTOR.TB.2:
U 0B03, 17,D815 ;12763         PORT.CONTROL [SEL.FIFO.A]                ;read FIFO A
U 0B04, 17,C015 ;12764         PORT.CONTROL [CLEAR.FIFO.CNTR]           ;clear FIFO A address counter
U 0B05, 90,9015 ;12765         MISC [SET.PORT.I.O]                      ;select IDC
U 0B06, 97,8C15 ;12766         PORT.READ PORT.ADRS[DATA.WORD]           ;read the header from FIFO A
U 0B07, 3B,3215 ;12767         MOV PORT TO LS[PORT0]                    ; and load that in
U 0B08, 37,3215 ;12768         MOV LS[PORT0] TO WR[0]                   ;  WR0 as received data
U 0B09, B7,C295 ;12769         MOV LS[#C000] TO WR[1]                   ;bad sector flag mask <15:14>
U 0B0A, 2F,C215 ;12770         AND WR[1] TO WR[0]                       ;look at UF & MF bits
                ;12771         CMP WR[0] WITH WR[1],                    ;good sector being read ??
U 0B0B, 26,40B5 ;12772          DT(LONG)&SET.ALU.CC
U 0B0C, 88,B129 ;12773         JMP.IF[EQL] TO [IDC.DONE.TB.4]           ;yes, good sector, go on
U 0B0D, B6,1F15 ;12774         MOV LS[T15] TO WR[2]                     ;no, bring in the time counter
                ;12775         BIT LS[BIT5] WITH WR[2],                 ;time expired ??
U 0B0E, D9,4B35 ;12776          DT(LONG)&SET.ALU.CC
U 0B0F, 08,B2B1 ;12777         JMP.IF[BIT.SET] TO [BAD.SECTOR.TB.2]     ;yes, do an error report
U 0B10, FF,1E15 ;12778         INC LS[T15]                              ;not yet, inc. time counter
U 0B11, 08,AFD4 ;12779         JMP [T.B.6]                              ;do another read header
                ;12780
                ;12781 ;Read the R80 header from FIFO A to verify if the heads are really at the desired cylinder :
                ;12782
                ;12783 IDC.DONE.TB.4:
U 0B12, 37,3215 ;12784         MOV LS[PORT0] TO WR[0]                   ;  WR0 as received data
U 0B13, 37,8895 ;12785         MOV LS[CYL.CNTR] TO WR[1]                ;expected cylinder = 0
U 0B14, 08,693C ;12786         JSR [CHECK.RESULT]                       ;heads at expected cylinder ??
U 0B15, 88,ADD4 ;12787         JMP [HEAD.SELECT.1]                      ;no, err and if LOE is set
U 0B16, 08,B174 ;12788         JMP [T.B.7]                              ;yes, next test
                ;12789
                ;12790 ;Update overheads for seeking to the next cylinder :
                ;12791
                ;12792 T.B.7:
U 0B17, B7,8815 ;12793         MOV LS[CYL.CNTR] TO WR[0]                ;fetch the cylinder counter
                ;12794         BIT LS[BIT9] WITH WR[0],                 ;has last tag bus bit <9> checked ?
U 0B18, 59,5235 ;12795          DT(LONG)&SET.ALU.CC
U 0B19, 88,B381 ;12796         JMP.IF[BIT.SET] TO [END.TB]              ;yes, test ends
U 0B1A, A3,C015 ;12797         ROL WR[0]                                ;no, shift tag bus 1 bit to left
U 0B1B, 3F,8815 ;12798         MOV WR[0] TO LS[CYL.CNTR]                ;and save current cylinder
U 0B1C, 88,ADD4 ;12799         JMP [HEAD.SELECT.1]                      ;seek to the next higher cylinder
                ;12800
                ;12801 BAD.SECTOR.TB.1:
```

```
U 0B1D, B7,9315   ;12802          MOV LS[#5] TO WR[2]                          ;load error number = 5
U 0B1E, 3E,8315   ;12803          MOV WR[2] TO LS[ERROR.NUMBER]
U 0B1F, B6,4F15   ;12804          MOV LS[R80] TO WR[2]                         ;fetch error module
U 0B20, BE,8D15   ;12805          MOV WR[2] TO LS[MODULE.NUM]
U 0B21, B7,C315   ;12806          MOV LS[#C000] TO WR[2]                       ;error mask for [CHECK.RESULT]
U 0B22, A0,C515   ;12807          COM WR[2]
U 0B23, BE,8B15   ;12808          MOV WR[2] TO LS[ERROR.MASK]
U 0B24, 36,7115   ;12809          MOV LS[OTHER.DATA] TO WR[2]                  ;CSR status for "OTHER DATA"
U 0B25, BE,8115   ;12810          MOV WR[2] TO LS[CONTROL.STATUS]
U 0B26, 37,3395   ;12811          MOV LS[PORTO] TO WR[3]
U 0B27, BE,8995   ;12812          MOV WR[3] TO LS[ADDRESS.DATA]
U 0B28, 08,693C   ;12813          JSR [CHECK.RESULT]
U 0B29, 08,AA74   ;12814          JMP [T.B.3]                                  ;err and if LOE is set
U 0B2A, 88,ADB4   ;12815          JMP [INIT.TB.1]
                  ;12816
                  ;12817 BAD.SECTOR.TB.2:
U 0B2B, B7,9515   ;12818          MOV LS[#6] TO WR[2]                          ;load error number = 6
U 0B2C, 3E,8315   ;12819          MOV WR[2] TO LS[ERROR.NUMBER]
U 0B2D, B6,4F15   ;12820          MOV LS[R80] TO WR[2]                         ;fetch error module
U 0B2E, BE,8D15   ;12821          MOV WR[2] TO LS[MODULE.NUM]
U 0B2F, B7,C315   ;12822          MOV LS[#C000] TO WR[2]                       ;error mask for [CHECK.RESULT]
U 0B30, A0,C515   ;12823          COM WR[2]
U 0B31, BE,8B15   ;12824          MOV WR[2] TO LS[ERROR.MASK]
U 0B32, 36,7115   ;12825          MOV LS[OTHER.DATA] TO WR[2]                  ;CSR status for "OTHER DATA"
U 0B33, BE,8115   ;12826          MOV WR[2] TO LS[CONTROL.STATUS]
U 0B34, 37,3395   ;12827          MOV LS[PORTO] TO WR[3]
U 0B35, BE,8995   ;12828          MOV WR[3] TO LS[ADDRESS.DATA]
U 0B36, 08,693C   ;12829          JSR [CHECK.RESULT]
U 0B37, 08,AFD4   ;12830          JMP [T.B.6]                                  ;err and if LOE is set
                  ;12831
                  ;12832 END.TB:                                               ;end of test
                  ;12833
```

```
;12834 .PAGE  "Test C  R80 INITIALIZATION SIGNAL TEST (not run in CRD mode)"
;12835 ;++
;12836 ;
;12837 ; Test Description:
;12838 ;
;12839 ;        Sending the R80 INITIALIZATION signal through CLEAR IDC PORT MICRO
;12840 ;        instruction from the cpu will clear head select of the DAR for a
;12841 ;        seek function and enable the motion control of the drive. This test
;12842 ;        will test such signal by using the first event. It will first select
;12843 ;        a head and then write to the IDC with a seek function; and it will
;12844 ;        wait for the function to complete and then clear IDC port instruction
;12845 ;        will be sent from the cpu. The head select DAR will then examined.
;12846 ;
;12847 ; Assumption:
;12848 ;
;12849 ;        Both CPU micros and IDC part I has been run successfully and
;12850 ;        drive 0 is a R80, up and spinning and ready to accept commands;
;12851 ;        R80 get status function is working; R80 recalibrate, seek and
;12852 ;        read header functions are working properly.
;12853 ;
;12854 ;        This test assumes "DISK DRIVE SIZING TEST" has been run prior to this
;12855 ;        test.
;12856 ;
;12857 ; Test steps:
;12858 ;
;12859 ;          1. Load LS[DR0.R80] and examine its content
;12860 ;          2. If value is zero, go to end of test
;12861 ;          3. Else initialize the first head select bit and set head select code
;12862 ;          4. Write DAR with the current format
;12863 ;          5. Write CSR with seek function bit set and select drive 0
;12864 ;          6. Wait for CRDY to set
;12865 ;          7. Send PORT instruction to clear IDC
;12866 ;          8. Read DAR back
;12867 ;          9. Check if the head select bits are all cleared
;12868 ;         10. If they are not; report ERROR 1
;12869 ;         11. Else check if all bits have been checked
;12870 ;         12. If so; then go to step 15
;12871 ;         13.        else select the next higher head select bit
;12872 ;         14. Go to step 4
;12873 ;         15. Test ends
;12874 ;
;12875 ; Error:
;12876 ;
;12877 ; Error 1 - Sending an initialize to R80 does not clear the head select bit
;12878 ;
;12879 ;             Printout:
;12880 ;                  EXPECTED       RECEIVED       MODULE
;12881 ;                  DAR (head      DAR (head      IDC
;12882 ;                  select bit     select bit)
;12883 ;                  = cleared)
;12884 ; Debug:
;12885 ;
;12886 ; ERROR 1: If the least significant head select bit did not flip to a
;12887 ;             different state, the R80 INITIALIZE may be at fault. Check the
;12888 ;             3_state differential drivers where R80 INITIALIZE signal comes
```

```
                   ;12889 ;         from and where CNTL TAG comes from; if needed, also the 74S174
                   ;12890 ;         flip flop for initialization. It can also be that the HD/CYL
                   ;12891 ;         TAG is at fault if other higher head select bits didn't change
                   ;12892 ;         state.
                   ;12893 ;
                   ;12894 ;--
                   ;12895
                   ;12896 T.C:
U 0B38, B6,5E15    ;12897        MOV LS[BEGIN.TEST] TO WR[0]              ;set bit 15 in WR[0] for
                   ;12898                                                 ; control and status word
U 0B39, 3E,8015    ;12899        MOV WR[0] TO LS[CONTROL.STATUS]          ;set bit 15 in control and
                   ;12900                                                 ; status word - bit 15
                   ;12901                                                 ; indicates start of test to
                   ;12902                                                 ; console
U 0B3A, 10,E015    ;12903        MISC [SET.CP.ATTN]                       ;issue CPU ATTN to console
                   ;12904
                   ;12905 WAIT.TC.0:
U 0B3B, 88,B3B4    ;12906        JMP [WAIT.TC.0]                          ;loop here waiting for
                   ;12907                                                 ; console to respond
                   ;12908
U 0B3C, 65,A015    ;12909        CLR LS[PREVIOUS.ERROR]                   ;clear previous error number
                   ;12910
U 0B3D, 17,CC15    ;12911        PORT.CONTROL [CLEAR.IDC]                 ;init. IDC
U 0B3E, 17,CC15    ;12912        PORT.CONTROL [CLEAR.IDC]
U 0B3F, 17,CC15    ;12913        PORT.CONTROL [CLEAR.IDC]
                   ;12914
                   ;12915 T.C.1:
U 0B40, B7,0C15    ;12916        MOV LS[DR0.R80] TO WR[0]                 ;load the R80 presence indicator
                   ;12917        TST WR[0],                               ;see if drive one has an R80
U 0B41, 20,0035    ;12918         DT(LONG)&SET.ALU.CC
U 0B42, 08,B649    ;12919        JMP.IF[EQL] TO [END.TC]                  ;no R80, go to end of test
                   ;12920
                   ;12921 ;R80 INITIALIZE issued through CLEAR IDC PORT MICRO_INSTRUCTION which will clear
                   ;12922 ; the head address register (HAR) of the head select DAR :
                   ;12923
                   ;12924 INIT.TC.1:
U 0B43, B6,4015    ;12925        MOV LS[BIT0] TO WR[0]                    ;bring in head select bit
U 0B44, 3E,0215    ;12926        MOV WR[0] TO LS[SAV.WR0]                 ;save it for later
                   ;12927
                   ;12928 T.C.2:
U 0B45, B6,0215    ;12929        MOV LS[SAV.WR0] TO WR[0]                 ;current head select bit
U 0B46, 47,5C15    ;12930        BIS LS[BIT14] TO WR[0]                   ;specify this is head select command
U 0B47, 97,A415    ;12931        PORT.WRITE PORT.ADRS[DAR] WITH WR[0]     ;DAR head select format
                   ;12932
U 0B48, 37,0415    ;12933        MOV LS[SEEK] TO WR[0]                    ;fetch the seek command
U 0B49, 47,9C15    ;12934        BIS LS[SETCSR.DS0] TO WR[0]              ;select drive zero
U 0B4A, 17,A015    ;12935        PORT.WRITE PORT.ADRS[CSR] WITH WR[0]     ;issue the head select command
                   ;12936
                   ;12937         ;*****************************************************************
                   ;12938         ;* REFER TO THE R80 SEEK IDC FUNCTION CODES AS LISTED IN TEST B *
                   ;12939         ;*****************************************************************
                   ;12940
                   ;12941 WAIT.IDC.TC.1:
U 0B4B, 88,700C    ;12942        JSR [WAIT.CSR.CRDY]                      ;see if CRDY is set
U 0B4C, 5B,001E    ;12943        SKIP.IF[SKIP]                            ;yes, proceed
```

```
U 0B4D, 08,B4B4   ;12944            JMP [WAIT.IDC.TC.1]                         ;wait some more
                  ;12945
                  ;12946 ;The following port instruction should initiate the R80 INITIALIZE input to the
                  ;12947 ; R80 disk drive :
                  ;12948
U 0B4E, 17,CC15   ;12949            PORT.CONTROL [CLEAR.IDC]
                  ;12950
                  ;12951 ;Set up error parameters for ERROR 1:
                  ;12952
U 0B4F, 36,4095   ;12953            MOV LS[#1] TO WR[1]                         ;set up error number 1
U 0B50, 3E,8295   ;12954            MOV WR[1] TO LS[ERROR.NUMBER]
U 0B51, B7,AE95   ;12955            MOV LS[#F] TO WR[1]
U 0B52, A0,C295   ;12956            COM WR[1]                                   ;form error mask for
U 0B53, BE,8A95   ;12957            MOV WR[1] TO LS[ERROR.MASK]                 ; head select bits <3:0>
U 0B54, 36,4A95   ;12958            MOV LS[IDC] TO WR[1]                        ;store IDC as failing
U 0B55, BE,8C95   ;12959            MOV WR[1] TO LS[MODULE.NUM]                 ; module
                  ;12960
                  ;12961 ;At this point, the head select bits of the DAR should be all reset :
                  ;12962
U 0B56, 90,9015   ;12963            MISC [SET.PORT.I.O]                         ;set from port to bus
U 0B57, 17,8415   ;12964            PORT.READ PORT.ADRS[DAR]                    ;read the DAR back
U 0B58, 3B,3215   ;12965            MOV PORT TO LS[PORTO]                       ; and load it
U 0B59, 37,3215   ;12966            MOV LS[PORTO] TO WR[0]                      ;  in as received data
U 0B5A, 2F,8295   ;12967            CLR WR[1]                                   ;expected data
U 0B5B, 08,693C   ;12968            JSR [CHECK.RESULT]                          ;are the head select bits reset ?
U 0B5C, 88,B454   ;12969            JMP [T.C.2]                                 ;err and if LOE is set
U 0B5D, 88,B5E4   ;12970            JMP [T.C.3]
                  ;12971
                  ;12972 ;Check if all head select bits are done :
                  ;12973
                  ;12974 T.C.3:
U 0B5E, B6,0215   ;12975            MOV LS[SAV.WRO] TO WR[0]                    ;bring in the current H.S. bit
                  ;12976            BIT LS[BIT3] WITH WR[0],                    ;last bit ??
U 0B5F, 59,4635   ;12977             DT(LONG)&SET.ALU.CC
U 0B60, 88,B641   ;12978            JMP.IF[BIT.SET] TO [END.TC]                 ;all done, test ends
                  ;12979
U 0B61, A3,C015   ;12980            ROL WR[0]                                   ;get to the next higher bit
U 0B62, 3E,0215   ;12981            MOV WR[0] TO LS[SAV.WRO]                    ;save for next pass
U 0B63, 88,B454   ;12982            JMP [T.C.2]                                 ;go do it
                  ;12983
                  ;12984 END.TC:                                                ;end of test
                  ;12985
```

```
;12986 .PAGE  "Test D  R80 WRITE AND READ DATA FUNCTION TEST (not run in CRD mode)"
;12987 ;++
;12988 ;
;12989 ; Test Description:
;12990 ;
;12991 ;          This test will write to the maintenance cylinder of the R80 ( i.e.
;12992 ;          cylinder 561 dec.) by writing bits 3 and 1 to the CSR and loading
;12993 ;          the data buffers with a data pattern. After XFER REQ is set, a read
;12994 ;          data function will be issued by writing bits 3 and 2 to the CSR and
;12995 ;          wait for another XFER REQ before reading the data pattern back from
;12996 ;          the data buffer. The data read will be compared with the one written
;12997 ;          to check for error.
;12998 ;
;12999 ; Assumption:
;13000 ;
;13001 ;          Both CPU micros and IDC part I has been run successfully and
;13002 ;          drive 0 is a R80, up and spinning and ready to accept commands;
;13003 ;          R80 get status function is working; R80 recalibrate, initialization,
;13004 ;          seek and read header functions are working properly.
;13005 ;
;13006 ;          This test assumes "DISK DRIVE SIZING TEST" has been run prior to this
;13007 ;          test.
;13008 ;
;13009 ; Test steps:
;13010 ;
;13011 ;           1. Load LS[DR0.R80] and examine its content
;13012 ;           2. If value is zero, then go to end of test
;13013 ;           3. Else seek to track 561
;13014 ;           4. Init. the sector to be written on
;13015 ;           5. Load FIFO A with a data pattern
;13016 ;           6. Load DAR with the selected sector, cylinder 561, head 3
;13017 ;           7. Invoke the write data function to the R80; CSR <3 & 1>
;13018 ;           8. Set up a time out loop waiting for XFER REQ to set
;13019 ;           9. Set up pertinent error parameters for ERROR 1
;13020 ;          10. If timeout, report the error
;13021 ;          11. Else check if the current sector is a good sector, if bad
;13022 ;          12.     Then see if it has tried out all 31 sector on this track, if so;
;13023 ;          13.             Then report ERROR 4
;13024 ;          14.             Else inc. to the next higher sector
;13025 ;          15.                     and go to step 6
;13026 ;          16.     Else load FIFO A with all zeros
;13027 ;          17. Load DAR with the selected sector and cylinder 561, head 3
;13028 ;          18. Invoke the read data function to the R80; CSR <3 & 2>
;13029 ;          19. Set up a time out loop waiting for XFER REQ to set
;13030 ;          20. Set up pertinent error parameters for ERROR 2
;13031 ;          21. If timeout, report the error
;13032 ;          22. Else read the data from FIFO A
;13033 ;          23. Set up pertinent error parameters for ERROR 3
;13034 ;          24. Compare it with the expected data by JSR[CHECK.RESULT]
;13035 ;          25. If data non-compare; report the error
;13036 ;          26. Else test ends
;13037 ;
;13038 ; Errors:
;13039 ;
;13040 ; ERROR 1 - Time out waiting for XFER REQ to set after a Write Data
```

```
                  ;13041 ;                 command has been issued to the IDC
                  ;13042 ;
                  ;13043 ; ERROR 2 - Time out waiting for XFER REQ to set after a Read Data
                  ;13044 ;                 command has been issued to the IDC
                  ;13045 ;
                  ;13046 ;         Printout:
                  ;13047 ;              EXPECTED       RECEIVED         MODULE
                  ;13048 ;              N/A            N/A              IDC
                  ;13049 ;
                  ;13050 ; ERROR 3 - Data non-compare after W/R operations
                  ;13051 ;
                  ;13052 ;         Printout:
                  ;13053 ;              EXPECTED       RECEIVED         MODULE
                  ;13054 ;              data           data             R80
                  ;13055 ;
                  ;13056 ; ERROR 4 - No good sector found on the current track
                  ;13057 ;
                  ;13058 ;         Printout:
                  ;13059 ;              EXPECTED       RECEIVED         MODULE
                  ;13060 ;              N/A            N/A              R80
                  ;13061 ;
                  ;13062 ; Debug:
                  ;13063 ;
                  ;13064 ; ERROR 1: Check DRDY signal, R80 SEC PLS, R80 READ CLK, R80 SERVO CLK, R80
                  ;13065 ;          DS CLK, SECTOR PLS, READ DATA & SYS CLK for fault or check the
                  ;13066 ;          operation of mux 74S158 of which all of the above signals come
                  ;13067 ;          from; and also check the header comparator circuit; and R80 WRITE
                  ;13068 ;          CLOCK & R80 WRITE DATA from the 3_state differential driver; WRITE
                  ;13069 ;          TAG from the multiplexer 74LS157.
                  ;13070 ;
                  ;13071 ; ERROR 2: Check DRDY signal, R80 SEC PLS, R80 READ CLK, R80 SERVO CLK, R80
                  ;13072 ;          DS CLK, SECTOR PLS, READ DATA & SYS CLK for fault or check the
                  ;13073 ;          operation of mux 74S158 of which all of the above signals come
                  ;13074 ;          from; and also check the header comparator circuit; R80 READ DATA
                  ;13075 ;          from the 3450 quad line receiver; READ TAG from the multiplexer
                  ;13076 ;          74LS157.
                  ;13077 ;
                  ;13078 ; ERROR 3: After the operation of the IDC function codes, check with the CSR
                  ;13079 ;          to see if OPI & DCK bits are set. If they are, then there might
                  ;13080 ;          have been a ECC/CRC error; check the CRC/ECC logic circuit.
                  ;13081 ;
                  ;13082 ; ERROR 4: It could not find a good sector on the disk at this particular
                  ;13083 ;          cylinder.
                  ;13084 ;
                  ;13085 ;--
                  ;13086
                  ;13087 T.D:
U 0B64, B6,5E15   ;13088          MOV LS[BEGIN.TEST] TO WR[0]          ;set bit 15 in WR[0] for
                  ;13089                                               ; control and status word
U 0B65, 3E,8015   ;13090          MOV WR[0] TO LS[CONTROL.STATUS]      ;set bit 15 in control and
                  ;13091                                               ; status word - bit 15
                  ;13092                                               ; indicates start of test to
                  ;13093                                               ; console
U 0B66, 10,E015   ;13094          MISC [SET.CP.ATTN]                   ;issue CPU ATTN to console
                  ;13095
                  ;13096 WAIT.TD.0:
U 0B67, 88,B674   ;13097         JMP [WAIT.TD.0]                           ;loop here waiting for
                  ;13098                                                   ; console to respond
                  ;13099
U 0B68, 65,A015   ;13100         CLR LS[PREVIOUS.ERROR]                    ;clear previous error number
                  ;13101                                                   ; in LS
U 0B69, 17,CC15   ;13102         PORT.CONTROL [CLEAR.IDC]                  ;init. IDC
U 0B6A, 17,CC15   ;13103         PORT.CONTROL [CLEAR.IDC]
U 0B6B, 17,CC15   ;13104         PORT.CONTROL [CLEAR.IDC]
                  ;13105
                  ;13106 T.D.1:
U 0B6C, B7,0C15   ;13107         MOV LS[DR0.R80] TO WR[0]                  ;load the R80 presence indicator
                  ;13108         TST WR[0],                                ;see if drive one has an R80
U 0B6D, 20,0035   ;13109          DT(LONG)&SET.ALU.CC
U 0B6E, 88,C049   ;13110         JMP.IF[EQL] TO [END.TD]                   ;no R80, go to end of test
                  ;13111
                  ;13112 ;In order to do a seek on the R80, a head select operation must be first initiated
                  ;13113 ; and it is then followed by the actual seek :
                  ;13114
                  ;13115 HEAD.SELECT.TD.1:
U 0B6F, 36,4215   ;13116         MOV LS[#2] TO WR[0]                       ;selecting the third head
U 0B70, 47,5C15   ;13117         BIS LS[BIT14] TO WR[0]                    ;set head select code
U 0B71, 97,A415   ;13118         PORT.WRITE PORT.ADRS[DAR] WITH WR[0]      ;load IDC DAR
                  ;13119
U 0B72, 37,0415   ;13120         MOV LS[SEEK] TO WR[0]                     ;fetch the seek command
U 0B73, 47,9C15   ;13121         BIS LS[SETCSR.DS0] TO WR[0]               ;select drive zero
U 0B74, 17,A015   ;13122         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;initiate the head select command
                  ;13123
                  ;13124 ;It is necessary to wait for CRDY before issuing another command or destroying the
                  ;13125 ; DAR :
                  ;13126
                  ;13127 WAIT.HEAD.SELECT.DONE.TD.1:
U 0B75, 88,700C   ;13128         JSR [WAIT.CSR.CRDY]                       ;check if CRDY is ready
U 0B76, 08,B784   ;13129         JMP [CLR.ATN.BITS.TD.1]                   ;yes, CRDY is set now
U 0B77, 88,B754   ;13130         JMP [WAIT.HEAD.SELECT.DONE.TD.1]          ;no, wait some more
                  ;13131
                  ;13132 ;This section of code is to insure that the proper attention bit is cleared after
                  ;13133 ; the implied head select :
                  ;13134
                  ;13135 CLR.ATN.BITS.TD.1:
U 0B78, 36,0A15   ;13136         MOV LS[T5.SUB] TO WR[0]                   ;bring in the CSR word
U 0B79, B6,6695   ;13137         MOV LS[BIT19] TO WR[1]                    ;CSR ATN3 bit
U 0B7A, 47,6495   ;13138         BIS LS[BIT18] TO WR[1]                    ;add CSR ATN2 bit
U 0B7B, AF,4215   ;13139         BIC WR[1] TO WR[0]                        ;reset ATN3:ATN2 bits
U 0B7C, 17,A015   ;13140         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;clear attn. bits in CSR
                  ;13141
                  ;13142 ;Start the actual seek to the cylinder 561 :
                  ;13143
                  ;13144 ACTUAL.R80.SEEK.TD:
U 0B7D, 37,B615   ;13145         MOV LS[#230] TO WR[0]                     ;cylinder address for DAR format
U 0B7E, 47,5A15   ;13146         BIS LS[BIT13] TO WR[0]                    ;specify this is a cyl. adr. word
U 0B7F, 97,C415   ;13147         PORT.CONTROL [RESET.BR]                   ;clear interrupt caused by head select
U 0B80, 97,A415   ;13148         PORT.WRITE PORT.ADRS[DAR] WITH WR[0]      ;write to the DAR
                  ;13149
U 0B81, 37,0415   ;13150         MOV LS[SEEK] TO WR[0]                     ;fetch the seek command
```

```
U 0B82, 47,9C15   ;13151          BIS LS[SETCSR.DSO] TO WR[0]                 ;select drive zero
U 0B83, 17,A015   ;13152          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]        ;start the R80 seek
                  ;13153
                  ;13154       ;*************************************************************************
                  ;13155       ;* REFER TO THE R80 SEEK IDC FUNCTION CODES AS LISTED IN THIS TEST ABOVE *
                  ;13156       ;*************************************************************************
                  ;13157
                  ;13158 WAIT.IDC.SEEK.TD.1:
U 0B84, 88,700C   ;13159          JSR [WAIT.CSR.CRDY]                         ;check if CRDY is set
U 0B85, 08,B874   ;13160          JMP [CHECK.DRDY.TD.1]                               ;yes, next test
U 0B86, 08,B844   ;13161          JMP [WAIT.IDC.SEEK.TD.1]                    ;wait some more
                  ;13162
                  ;13163 ;Check if the R80 is ready to take the next operation :
                  ;13164
                  ;13165 CHECK.DRDY.TD.1:
U 0B87, 08,708C   ;13166          JSR [WAIT.CSR.DRDY]                         ;wait for DRDY to set
U 0B88, 36,9C15   ;13167          MOV LS[ZERO] TO WR[0]                       ;fetch zero
U 0B89, BF,8015   ;13168          MOV WR[0] TO LS[R80.SECTOR]                 ;init. sector select for DAR
                  ;13169
                  ;13170 ;Fill every address (total = 512) of FIFO A with a fixed data byte pattern :
                  ;13171
                  ;13172 T.D.2:
U 0B8A, 2F,8295   ;13173          CLR WR[1]                                   ;FIFO A address counter
U 0B8B, 17,D815   ;13174          PORT.CONTROL [SEL.FIFO.A]                   ;port select FIFO A
U 0B8C, 17,C015   ;13175          PORT.CONTROL [CLEAR.FIFO.CNTR]              ;init. FIFO counter
U 0B8D, B6,9815   ;13176          MOV LS[@AAAAAAAA] TO WR[0]                  ;load data pattern
                  ;13177                                                      ;only the lowest data byte is
                  ;13178                                                      ; needed for this operation
                  ;13179 FILL.FIFO.TD.1:
U 0B8E, 97,A815   ;13180          PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;start writing one byte
U 0B8F, 20,4295   ;13181          INC WR[1]                                   ;inc. the address counter
                  ;13182          BIT WR[1] WITH LS[BIT10],                   ;see if it gets to 400 hex yet
U 0B90, D9,54B5   ;13183           DT(LONG)&SET.ALU.CC
U 0B91, 88,B8E9   ;13184          JMP.IF[BITS.CLR] TO [FILL.FIFO.TD.1]        ;not yet, do the next byte
                  ;13185
                  ;13186
                  ;13187 ;Now that FIFO A is full, start write data function to R80 :
                  ;13188 ;The disk address selected is head 3, sector 0 and cylinder 561 dec.
                  ;13189
                  ;13190 T.D.4:
U 0B92, B7,AC15   ;13191          MOV LS[@00046000] TO WR[0]                  ;select cyl. 561
U 0B93, C7,4C15   ;13192          BIS LS[BIT6] TO WR[0]                       ;select head 3
U 0B94, B7,8095   ;13193          MOV LS[R80.SECTOR] TO WR[1]                 ;load current selected sector
U 0B95, AE,C215   ;13194          BIS WR[1] TO WR[0]
U 0B96, 97,A415   ;13195          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]        ;load DAR for a write data function
                  ;13196
U 0B97, 37,0815   ;13197          MOV LS[WRITE.DATA] TO WR[0]                 ;select write data function
U 0B98, 47,9C15   ;13198          BIS LS[SETCSR.DSO] TO WR[0]                 ;select drive 0
U 0B99, 17,A015   ;13199          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]        ;write to the disk
                  ;13200
                  ;13201 ;*** R80 Write Data IDC function codes : ***
                  ;13202
                  ;13203 ;-----------------------------------------------------------------
                  ;13204 ;=1
                  ;13205 ;
```

```
;13206 ;TEST.OR.RUN:
;13207 ;
;13208 ;        BEN2/MAINT,                 ;IS THAT YOU, VICKIE?
;13209 ;        NAD/DRIVE.TYPE
;13210 ;
;13211 ;----------------------------------------------------
;13212 ;=0**
;13213 ;
;13214 ;DRIVE.TYPE:
;13215 ;
;13216 ;        BEN0/R80,                   ;WHICH TYPE OF DRIVE WAS SELECTED?
;13217 ;        NAD/RL02.IFORK
;13218 ;
;13219 ;----------------------------------------------------
;13220 ;=1
;13221 ;
;13222 ;R80.IFORK:
;13223 ;
;13224 ;        UCMD/SET.FIFOA,
;13225 ;        FUNC/CLR.ECC.ERROR,         ; Enable clear of ECC/CRC errors for new function
;13226 ;        BEN0/F0,                    ;8 WAY INSTRUCTION FORK.
;13227 ;        BEN1/F1,                    ;BRANCH ON F<2:0> IN THE CSR.
;13228 ;        BEN2/F2,
;13229 ;        NAD/@NOP.CMD
;13230 ;
;13231 ;----------------------------------------------------
;13232 ;=101
;13233 ;
;13234 ;@WRT.DAT.CMD:
;13235 ;
;13236 ;        NAD/@WAIT.DRDY1             ;START OF THE R80 WRITE DATA COMMAND
;13237 ;
;13238 ;----------------------------------------------------
;13239 ;=0
;13240 ;
;13241 ;@WAIT.DRDY1:
;13242 ;
;13243 ;        BEN0/DRIVE.RDY,             ;WAIT FOR DRIVE READY
;13244 ;        NAD/@WAIT.DRDY1
;13245 ;
;13246 ;----------------------------------------------------
;13247 ;=1
;13248 ;
;13249 ;@CHGCLK1:
;13250 ;
;13251 ;        ENB.SYS.CLK,                ;USE R80 SYSTEM CLOCK
;13252 ;        NAD/@WAIT.NOSEC.PLS1
;13253 ;
;13254 ;----------------------------------------------------
;13255 ;=1*
;13256 ;
;13257 ;@WAIT.NOSEC.PLS1:
;13258 ;
;13259 ;        BEN1/SECTOR.PLS,            ;PREPARE TO LOOK FOR THE LEADING EDGE
;13260 ;        NAD/@WAIT.NOSEC.PLS1        ;OF SECTOR PULSE.
```

```
;13261 ;
;13262 ;-----------------------------------------------------------------------
;13263 ;=0*
;13264 ;
;13265 ;@NOSEC.PLS1:
;13266 ;
;13267 ;       NAD/@WAIT.SEC.PLS1      ;WE ARE NOT IN SECTOR PULSE AREA.
;13268 ;
;13269 ;-----------------------------------------------------------------------
;13270 ;=0*
;13271 ;
;13272 ;@WAIT.SEC.PLS1:
;13273 ;
;13274 ;       BEN1/SECTOR.PLS,        ;WAIT FOR LEADING EDGE OF SEC PLS
;13275 ;       NAD/@WAIT.SEC.PLS1
;13276 ;
;13277 ;-----------------------------------------------------------------------
;13278 ;=1*
;13279 ;
;13280 ;@SEC.PLS.FND1:
;13281 ;
;13282 ;       LOAD.LOOP.CNTR.[0C5],   ;SECTOR PULSE HAS BEEN FOUND.
;13283 ;       NAD/@WAIT.60.TICKS1
;13284 ;
;13285 ;-----------------------------------------------------------------------
;13286 ;=0
;13287 ;
;13288 ;@WAIT.60.TICKS1:
;13289 ;
;13290 ;       INCR.LPCNTR/ON,         ;DELAY BEFORE ASSERTING READ GATE
;13291 ;       TAG.STROBE/ON,
;13292 ;       BEN0/CNTR.OVFLW,
;13293 ;       NAD/@WAIT.60.TICKS1
;13294 ;
;13295 ;-----------------------------------------------------------------------
;13296 ;=1
;13297 ;
;13298 ;@SETUP.LOOP.LOCK1:
;13299 ;
;13300 ;       LOOP.LOCK/ON,           ;ALLOW PHASE LOCK LOOP TO SYNC ON
;13301 ;       TAG.STROBE/ON,          ;ALL ZEROS PATTERN
;13302 ;       FUNC/CLR.ECC,
;13303 ;       LOAD.LOOP.CNTR.[0A9]
;13304 ;
;13305 ;-----------------------------------------------------------------------
;13306 ;=0
;13307 ;
;13308 ;@WAIT.LOOP.LOCK1:
;13309 ;
;13310 ;       LOOP.LOCK/ON,           ;WAIT FOR 88 TICKS AFTER READ GATE
;13311 ;       TAG.STROBE/ON,
;13312 ;       UCON/CLR.MATCH,
;13313 ;       INCR.LPCNTR/ON,
;13314 ;       BEN0/CNTR.OVFLW,
;13315 ;       NAD/@WAIT.LOOP.LOCK1
```

```
;13316 ;
;13317 ;--------------------------------------------------------------------
;13318 ;=1
;13319 ;
;13320 ;#SETUP.RD.HDR1:
;13321 ;
;13322 ;        LOOP.LOCK/ON,                  ;SETUP INITIAL CONDITIONS
;13323 ;        TAG.STROBE/ON,                 ;FOR READING HEADER.
;13324 ;        FUNC/CLR.DSR,
;13325 ;        ENB.CLR.BC/ON,
;13326 ;        LOAD.LOOP.CNTR.[0E1]
;13327 ;
;13328 ;--------------------------------------------------------------------
;13329 ;
;13330 ;#CHGCLK1A:
;13331 ;
;13332 ;        ENB.DSK.CLK,                   ;THE MICROSEQUENCER WILL STALL
;13333 ;        LOOP.LOCK/ON,                  ;UNTIL SYNC SEEN IS SET
;13334 ;        TAG.STROBE/ON,                 ;(SYNC BYTE FOUND)
;13335 ;        ENB.CLR.BC/ON,
;13336 ;        GEN.SYNC.[19]
;13337 ;
;13338 ;--------------------------------------------------------------------
;13339 ;=0
;13340 ;
;13341 ;#READ.HDR1:
;13342 ;
;13343 ;        LOOP.LOCK/ON,                  ;SHIFT IN 32 HEADER BITS AND
;13344 ;        TAG.STROBE/ON,                 ;COMPARE WITH SERIAL DAR
;13345 ;        DATA.FIELD/ON,
;13346 ;        ENB.CLR.BC/ON,
;13347 ;        INCR.LPCNTR/ON,
;13348 ;        BENO/CNTR.OVFLW,
;13349 ;        NAD/#READ.HDR1
;13350 ;
;13351 ;--------------------------------------------------------------------
;13352 ;=1
;13353 ;
;13354 ;#TEST.HDR.MATCH1:
;13355 ;
;13356 ;        UCON/TEST.MATCH,
;13357 ;        LOOP.LOCK/ON,                  ;HEADER HAS BEEN READ.
;13358 ;        TAG.STROBE/ON,                 ;TURN ON TEST MATCH, OFF ENB CLR BC.
;13359 ;        DATA.FIELD/ON,                 ;PREPARE TO READ THE CRC WORD.
;13360 ;        NAD/#BRANCH.HDR.MATCH1
;13361 ;
;13362 ;--------------------------------------------------------------------
;13363 ;
;13364 ;#BRANCH.HDR.MATCH1:
;13365 ;
;13366 ;        LOOP.LOCK/ON,
;13367 ;        TAG.STROBE/ON,
;13368 ;        CHECK.FIELD/ON,                ;RESULTS OF HDR COMP NOW IN MISMATCH.
;13369 ;        BEN2/MISMATCH.L,
;13370 ;        NAD/#HDR.COMP1
```

```
;
;-----------------------------------------------------------------
;=1**
;
;#HDR.COMP1:
;
;       LOOP.LOCK/ON,           ;HEADER COMPARE
;       TAG.STROBE/ON,
;       CHECK.FIELD/ON,         ;CONTINUE TO READ THE CRC WORD.
;       LOAD.LOOP.CNTR.[OF3],
;       BENO/CSSE,              ;CHECK FOR SKIP SECTOR ERROR.
;       NAD/#NOT.SSE1
;
;-----------------------------------------------------------------
;=0
;
;#NOT.SSE1:
;
;       LOOP.LOCK/ON,
;       TAG.STROBE/ON,
;       CHECK.FIELD/ON,         ;CONTINUE TO READ THE CRC WORD.
;       NAD/#RD.CRC.WD1
;
;-----------------------------------------------------------------
;=0
;
;#RD.CRC.WD1:
;
;       LOOP.LOCK/ON,           ;CONTINUE TO SHIFT IN THE 16 BIT
;       TAG.STROBE/ON,          ;CRC WORD.
;       CHECK.FIELD/ON,
;       INCR.LPCNTR/ON,
;       BENO/CNTR.OVFLW,
;       NAD/#RD.CRC.WD1
;
;-----------------------------------------------------------------
;=1
;
;#CHECK.CRC1:
;
;       LOOP.LOCK/ON,           ;DROP CFLD. CLOCK ECC (CRC) FLOP.
;       TAG.STROBE/ON,
;       UCON/CHECK.ECC
;
;-----------------------------------------------------------------
; This is a new state for U14IDC to allow DC007 time to
; settle error bits.
;
;#CHECK.CRC2:
;
;       LOOP.LOCK/ON,           ;DROP CFLD. CLOCK ECC (CRC) FLOP.
;       TAG.STROBE/ON,
;       UCON/CHECK.ECC
;
;-----------------------------------------------------------------
```

```
;
;#TEST.FOR.ERR1:
;
;       LOOP.LOCK/ON,           ;WAS THERE A CRC ERROR?
;       TAG.STROBE/ON,
;       BEN1/ERROR.L
;
;-------------------------------------------------------------------
;=1*
;
;#NOT.CRC.ERR1:
;
;       LOOP.LOCK/ON,           ;CLEAR DC007
;       TAG.STROBE/ON,
;       FUNC/CLR.ECC,
;       BEN2/F2,                ;BRANCH ON FUNCTION BITS INTO WRT CHK,
;       BEN1/F1,                ;RD DATA OR WRT DATA FLOWS.
;       NAD/#BRANCH.WC.CMD1
;
;-------------------------------------------------------------------
;=10*
;
;#BRANCH.WRT.CMD5:
;
;       LOOP.LOCK/ON,           ;CONTINUE VIA WRITE DATA FLOWS.
;       TAG.STROBE/ON,
;       BEN1/FIFO.OVFLW.L,
;       NAD/#CHGCLK5B
;
;-------------------------------------------------------------------
;=0*
;
;#CHGCLK5B:
;
;       LOOP.LOCK/ON,           ;EVERYTHING IS GO FOR WRITING DATA
;       TAG.STROBE/ON,          ;INTO THIS SECTOR. RUN OFF SYSTEM CLOCK
;       ENB.SYS.CLK,            ;THRU HEADER POSTAMBLE TO SPLICE.
;       LOAD.LOOP.CNTR.[0F9],
;       NAD/#HDR.POST5
;
;-------------------------------------------------------------------
;=0
;
;#HDR.POST5:
;
;       TAG.STROBE/ON,          ;DROP READ GATE.
;       INCR.LPCNTR/ON,         ;SHIFT OVER POSTAMBLE.
;       BEN0/CNTR.OVFLW,
;       NAD/#HDR.POST5
;
;-------------------------------------------------------------------
;=1
;
;#SETUP.WRT.DAT5:
;
```

```
;13481 ;        DISK.WRITE/ON,
;13482 ;        TAG.STROBE/ON,
;13483 ;        LOAD.LOOP.CNTR.[87],     ;PREPARE TO WRITE PREAMBLE
;13484 ;        CLR.FIFO.ADR,
;13485 ;        FUNC/CLR.DSR
;13486 ;
;13487 ;----------------------------------------------------------------------
;13488 ;=0
;13489 ;
;13490 ;#WRT.DAT.PRE5:
;13491 ;
;13492 ;        DISK.WRITE/ON,           ;WRITE 15 BYTES OF ZEROS (120 BITS)
;13493 ;        WRITE.GATE/ON,
;13494 ;        TAG.STROBE/ON,
;13495 ;        INCR.LPCNTR/ON,
;13496 ;        BENO/CNTR.OVFLW,
;13497 ;        NAD/#WRT.DAT.PRE5
;13498 ;
;13499 ;----------------------------------------------------------------------
;13500 ;=1
;13501 ;
;13502 ;#LOAD.SYNC.BYTE5:
;13503 ;
;13504 ;        DISK.WRITE/ON,           ;WRITE LAST ZERO BIT
;13505 ;        WRITE.GATE/ON,
;13506 ;        TAG.STROBE/ON,
;13507 ;        ENB.SYNC.[19],           ;LOAD SYNC BYTE 00011001
;13508 ;        LOAD.DSR/ON
;13509 ;
;13510 ;----------------------------------------------------------------------
;13511 ;
;13512 ;#WRT.SYNC.BYTE5:
;13513 ;
;13514 ;        DISK.WRITE/ON,           ;WRITE SYNC BIT 0
;13515 ;        WRITE.GATE/ON,
;13516 ;        TAG.STROBE/ON,
;13517 ;        LOAD.LOOP.CNTR.[0FA]
;13518 ;
;13519 ;----------------------------------------------------------------------
;13520 ;=0
;13521 ;
;13522 ;#WRT.SYNC.BYTE5A:
;13523 ;
;13524 ;        DISK.WRITE/ON,           ;WRITE SYNC BITS 1 - 6
;13525 ;        WRITE.GATE/ON,
;13526 ;        TAG.STROBE/ON,
;13527 ;        INCR.LPCNTR/ON,
;13528 ;        BENO/CNTR.OVFLW,
;13529 ;        NAD/#WRT.SYNC.BYTE5A
;13530 ;
;13531 ;----------------------------------------------------------------------
;13532 ;=1
;13533 ;
;13534 ;#LD.1ST.DAT.BYTE5:
;13535 ;
```

```
;13536 ;        DISK.WRITE/ON,              ;WRITE SYNC BIT 7.
;13537 ;        WRITE.GATE/ON,
;13538 ;        TAG.STROBE/ON,
;13539 ;        ENB.FIFO/ON,
;13540 ;        LOAD.DSR/ON,                ;LOAD FIRST DATA BYTE.
;13541 ;        MODE/ECC,
;13542 ;        INCR.FIFO.CNTR/ON,
;13543 ;        LOAD.LOOP.CNTR.[OF9],
;13544 ;        NAD/#WRITE.DATA5
;13545 ;
;13546 ;--------------------------------------------------------------------
;13547 ;=10
;13548 ;
;13549 ;#WRITE.DATA5:
;13550 ;
;13551 ;        DISK.WRITE/ON,              ;SHIFT OUT 7 DATA BITS (0 - 6)
;13552 ;        WRITE.GATE/ON,
;13553 ;        TAG.STROBE/ON,
;13554 ;        DATA.FIELD/ON,
;13555 ;        MODE/ECC,
;13556 ;        INCR.LPCNTR/ON,
;13557 ;        BENO/CNTR.OVFLW,
;13558 ;        NAD/#WRITE.DATA5
;13559 ;
;13560 ;--------------------------------------------------------------------
;13561 ;=11
;13562 ;
;13563 ;#LD.NEXT.DAT.BYTE5:
;13564 ;
;13565 ;        DISK.WRITE/ON,              ;WRITE BIT 7 (LAST BIT IN BYTE)
;13566 ;        WRITE.GATE/ON,
;13567 ;        TAG.STROBE/ON,
;13568 ;        DATA.FIELD/ON,
;13569 ;        MODE/ECC,
;13570 ;        ENB.FIFO/ON,
;13571 ;        LOAD.DSR/ON,                ;LOAD NEXT DATA BYTE.
;13572 ;        LOAD.LOOP.CNTR.[OF9],
;13573 ;        INCR.FIFO.CNTR/ON,
;13574 ;        BEN1/FIFO.MAX.L,
;13575 ;        NAD/#WRITE.DATA5
;13576 ;
;13577 ;--------------------------------------------------------------------
;13578 ;=00
;13579 ;
;13580 ;#WRT.LAST.DAT.BYTE5:
;13581 ;
;13582 ;        DISK.WRITE/ON,              ;WRITE LAST DATA BYTE (BITS 0 - 6)
;13583 ;        WRITE.GATE/ON,
;13584 ;        TAG.STROBE/ON,
;13585 ;        DATA.FIELD/ON,
;13586 ;        MODE/ECC,
;13587 ;        INCR.LPCNTR/ON,
;13588 ;        BENO/CNTR.OVFLW,
;13589 ;        NAD/#WRT.LAST.DAT.BYTE5
;13590 ;
```

```
;------------------------------------------------------------------------------
;=01
;
;#WRT.LAST.DAT.BIT5:
;
;       DISK.WRITE/ON,          ;WRITE LAST DATA BIT
;       WRITE.GATE/ON,
;       TAG.STROBE/ON,
;       DATA.FIELD/ON,
;       MODE/ECC,
;       LOAD.LOOP.CNTR.[0E1],
;       NAD/#WRT.DAT.ECC5
;
;------------------------------------------------------------------------------
;=0
;
;#WRT.DAT.ECC5:
;
;       DISK.WRITE/ON,          ;WRITE 32 BIT ECC
;       WRITE.GATE/ON,
;       TAG.STROBE/ON,
;       CHECK.FIELD/ON,
;       MODE/ECC,
;       INCR.LPCNTR/ON,
;       BENO/CNTR.OVFLW,
;       NAD/#WRT.DAT.ECC5
;
;------------------------------------------------------------------------------
;=1
;
;#WRT.DAT.ECC5A:
;
;       DISK.WRITE/ON,          ;WRITE LAST BIT OF ECC.
;       WRITE.GATE/ON,
;       TAG.STROBE/ON,
;       CHECK.FIELD/ON,
;       MODE/ECC,
;       LOAD.LOOP.CNTR.[0F0]
;
;------------------------------------------------------------------------------
;=0
;
;#WRT.DAT.POST5:
;
;       DISK.WRITE/ON,          ;WRITE 16 ZEROS OF POSTAMBLE.
;       WRITE.GATE/ON,
;       TAG.STROBE/ON,
;       MODE/ECC,
;       INCR.LPCNTR/ON,
;       BENO/CNTR.OVFLW,
;       NAD/#WRT.DAT.POST5
;
;------------------------------------------------------------------------------
;=1
;
```

```
;13646 ;#CHGCLK5C:
;13647 ;
;13648 ;        ENB.CPU.CLK                 ;SWITCH FOR FAST INTR OR TERM.
;13649 ;
;13650 ;----------------------------------------------------------------------
;13651 ;
;13652 ;#SET.XFER.RQST5:
;13653 ;
;13654 ;        CLR.FIFO.ADR,               ;CLEAR FIFO COUNTER FOR CPU.
;13655 ;        UCMD/SET.XFER.RQST,
;13656 ;        NAD/#WAIT.XFER.GRANT5
;13657 ;
;13658 ;----------------------------------------------------------------------
;13659 ;=1*
;13660 ;
;13661 ;#WAIT.XFER.GRANT5:
;13662 ;
;13663 ;        BEN1/XFER.REQ,              ;CPU CLEARS CRDY IF LAST SECTOR.
;13664 ;        NAD/#WAIT.XFER.GRANT5
;13665 ;
;13666 ;----------------------------------------------------------------------
;13667 ;=0
;13668 ;
;13669 ;#XFER.DONE5:
;13670 ;
;13671 ;        UCMD/SET.CRDY,
;13672 ;        INCR.DAR/ON,
;13673 ;        FUNC/SET.INT,               ;CPU SIGNALLED THAT XFER COMPLETED.
;13674 ;        NAD/SET.INT.AND.IDLE
;13675 ;
;13676 ;----------------------------------------------------------------------
;13677 ;=111111110
;13678 ;
;13679 ;SET.INT.AND.IDLE:
;13680 ;
;13681 ;        FUNC/SET.INT,               ; The purpose of this state is to give CRDY one
;13682 ;        NAD/IDLE                    ; state to reset before asserting interrupt.
;13683 ;                                    ; CRDY is now used in the INT equation.
;13684 ;
;13685 ;----------------------------------------------------------------------
;13686 ;=1
;13687 ;
;13688 ;RTN.TO.IDLE:
;13689 ;IDLE:
;13690 ;
;13691 ;        BEN0/CRDY.L,                ;LOOK FOR GO (CRDY L)
;13692 ;        NAD/SETUP.DRV.CHK
;13693 ;
;13694 ;----------------------------------------------------------------------
;13695 ;=0
;13696 ;
;13697 ;SETUP.DRV.CHK:
;13698 ;
;13699 ;        LOAD.LOOP.CNTR.[0FB],       ;COUNTER WILL BE USED
;13700 ;        NAD/SEL.NXT.DRV             ;TO SEQUENCE THRU DRIVES.
```

```
;13701 ;
;13702 ;-------------------------------------------------------------------
;13703 ;
;13704 ;SEL.NXT.DRV:
;13705 ;
;13706 ;       SEL.DRV/UPROG,
;13707 ;       BENO/CNTR.OVFLW,
;13708 ;       NAD/CHK.DRV.STATUS
;13709 ;
;13710 ;-------------------------------------------------------------------
;13711 ;=0
;13712 ;
;13713 ;CHK.DRV.STATUS:
;13714 ;       ; Give drive status time to settle down.
;13715 ;
;13716 ;       SEL.DRV/UPROG,          ;SELECT DRIVE
;13717 ;       NAD/DRV.STATUS.DISPATCH
;13718 ;
;13719 ;-------------------------------------------------------------------
;13720 ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
;13721 ;
;13722 ;DRV.STATUS.DISPATCH:
;13723 ;       SEL.DRV/UPROG,          ;SELECT DRIVE
;13724 ;       BEN1/DRIVE.ONLINE,      ;AND BRANCH ON STATUS.
;13725 ;       BENO/DRIVE.RDY,
;13726 ;       NAD/-ERR.-OL.-RDY
;13727 ;
;13728 ;
;13729 ;-------------------------------------------------------------------
;13730 ;=011
;13731 ;
;13732 ;-ERR.OL.RDY:
;13733 ;
;13734 ;       SEL.DRV/UPROG,          ;NOP
;13735 ;       INCR.LPCNTR/ON,         ;ADD 1 TO DRIVE SELECT.
;13736 ;       NAD/SEL.NXT.DRV
;13737 ;
;13738 ;-------------------------------------------------------------------
;13739 ;
;13740 ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
;13741 ;
;13742 ;               UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
;13743 ;
;13744 ;-------------------------------------------------------------------
;13745 ;
;13746 ;=1
;13747 ;
;13748 ;RTN.TO.IDLE:
;13749 ;IDLE:
;13750 ;
;13751 ;       BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
;13752 ;       NAD/SETUP.DRV.CHK
;13753 ;
;13754 ;-------------------------------------------------------------------
;13755
```

```
                      ;Set uo parameters for timeout error report :

U 0B9A, 36,4095               MOV LS[#1] TO WR[1]
U 0B9B, 3E,8295               MOV WR[1] TO LS[ERROR.NUMBER]         ;set up error number = 1 in LS
U 0B9C, E5,8A15               CLR LS[ERROR.MASK]                    ;no error mask used here
U 0B9D, 36,4A95               MOV LS[IDC] TO WR[1]                  ;store module code in LS to
U 0B9E, BE,8C95               MOV WR[1] TO LS[MODULE.NUM]           ; modules
U 0B9F, 36,6E95               MOV LS[NA.EXP.REC] TO WR[1]           ;put "N/A" for printout data
U 0BA0, BE,8095               MOV WR[1] TO LS[CONTROL.STATUS]

U 0BA1, 2F,8015               CLR WR[0]                             ;Initialize time-out counter
U 0BA2, BF,8C15               MOV WR[0] TO LS[TIMEOUT.CNTR]

                      WAIT.IDC.TD.1:
U 0BA3, 37,8C15               MOV LS[TIMEOUT.CNTR] TO WR[0]         ;load the current time count
U 0BA4, 37,9095               MOV LS[WAIT.TIME1] TO WR[1]           ;time limit on write data
                              CMP WR[0] WITH WR[1],                 ;timeout yet ???
U 0BA5, 26,40B5                DT(LONG)&SET.ALU.CC
U 0BA6, 88,BAC9               JMP.IF[EQL] TO [TD.ERR.1]             ;yes, timeout on write data
U 0BA7, 20,4015               INC WR[0]                             ;inc. time count
U 0BA8, BF,8C15               MOV WR[0] TO LS[TIMEOUT.CNTR]
U 0BA9, DB,001A               SKIP.IF[NO.FAST.INTERRUPT]            ;is XFER REQ set ???
U 0BAA, 08,BB14               JMP [IDC.DONE.TD.1]                   ;yes, XFER REQ is set now
U 0BAB, 08,BA34               JMP [WAIT.IDC.TD.1]                   ;no, wait some more

                      ;Force an error in the CHECK.RESULT routine for this timeout :

                      TD.ERR.1:
U 0BAC, 2F,8015               CLR WR[0]                             ;set unequal values
U 0BAD, 36,4095               MOV LS[#1] TO WR[1]                   ; between WR0 and WR1
U 0BAE, 08,693C               JSR [CHECK.RESULT]                    ;force an error
U 0BAF, 88,B924               JMP [T.D.4]                           ;err and if LOE is set
U 0BB0, 08,BB14               JMP [IDC.DONE.TD.1]

                      IDC.DONE.TD.1:
U 0BB1, 90,FA15               MISC2 [TRANSFER.GRANT]                ;clear the XFER REQ
U 0BB2, 90,9015               MISC [SET.PORT.I.O]                   ;select from port to bus
U 0BB3, 97,8015               PORT.READ PORT.ADRS[CSR]              ;read the CSR
U 0BB4, 3B,3215               MOV PORT TO LS[PORTO]                 ;save the CSR word
U 0BB5, 37,3215               MOV LS[PORTO] TO WR[0]                ;load CSR word
U 0BB6, 37,C095               MOV LS[#C00] TO WR[1]                 ;mask for OPI & DCK bits
                              AND WR[1] TO WR[0],                   ;mask out these bits
U 0BB7, AF,C235                DT(LONG)&SET.ALU.CC                  ;CRC header error ??
U 0BB8, 88,BC09               JMP.IF[EQL] TO [T.D.5]                ;no, go on
U 0BB9, 37,8015               MOV LS[R80.SECTOR] TO WR[0]           ;yes, fetch sector select
U 0BBA, B7,1A95               MOV LS[#1E] TO WR[1]                  ;highest sector on a R80 track
                              CMP WR[0] WITH WR[1],                 ;tried all sector yet ??
U 0BBB, 26,40B5                DT(LONG)&SET.ALU.CC
U 0BBC, 88,BF99               JMP.IF[EQL] TO [BAD.SECTOR.TD.1]      ;report bad sector error
U 0BBD, 20,4015               INC WR[0]                             ;try the next higher sector
U 0BBE, BF,8015               MOV WR[0] TO LS[R80.SECTOR]
U 0BBF, 88,B8A4               JMP [T.D.2]                           ;go write data in this sector

                      ;Write off data in FIFO A with zeros before doing a read data function :

```

```
                  ;13811 T.D.5:
U 0BC0, 2F,8295   ;13812         CLR WR[1]                                ;FIFO A address counter
U 0BC1, 17,D815   ;13813         PORT.CONTROL [SEL.FIFO.A]                ;port select FIFO A
U 0BC2, 17,C015   ;13814         PORT.CONTROL [CLEAR.FIFO.CNTR]           ;init. FIFO counter
U 0BC3, 2F,8015   ;13815         CLR WR[0]                                ;data = 0
                  ;13816
                  ;13817 FILL.FIFO.TD.2:
U 0BC4, 97,A815   ;13818         PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;start writing one byte
U 0BC5, 20,4295   ;13819         INC WR[1]                                ;inc. the address counter
                  ;13820         BIT WR[1] WITH LS[BIT10],                ;see if it gets to 400 hex yet
U 0BC6, D9,54B5   ;13821          DT(LONG)&SET.ALU.CC
U 0BC7, 08,BC49   ;13822         JMP.IF[BITS.CLR] TO [FILL.FIFO.TD.2]     ;not yet, do the next byte
                  ;13823
                  ;13824 ;Now that FIFO A is refilled with zeros, start read data function to
                  ;13825 ; R80. The disk address selected is the same as the write data's :
                  ;13826
                  ;13827 T.D.7:
U 0BC8, B7,AC15   ;13828         MOV LS[#00046000] TO WR[0]               ;select cyl. 561
U 0BC9, C7,4C15   ;13829         BIS LS[BIT6] TO WR[0]                    ;select head 3
U 0BCA, B7,8095   ;13830         MOV LS[R80.SECTOR] TO WR[1]              ;load current selected sector
U 0BCB, AE,C215   ;13831         BIS WR[1] TO WR[0]
U 0BCC, 97,A415   ;13832         PORT.WRITE PORT.ADRS[DAR] WITH WR[0]     ;load DAR for a write data function
                  ;13833
U 0BCD, B7,0A15   ;13834         MOV LS[READ.DATA] TO WR[0]               ;select read data function
U 0BCE, 47,9C15   ;13835         BIS LS[SETCSR.DS0] TO WR[0]              ;select drive 0
U 0BCF, 17,A015   ;13836         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]     ;read from the disk
                  ;13837
                  ;13838 ;*** R80 Read Data IDC function codes : ***
                  ;13839
                  ;13840 ;-----------------------------------------------------------------
                  ;13841 ;=1
                  ;13842 ;
                  ;13843 ;TEST.OR.RUN:
                  ;13844 ;
                  ;13845 ;       BEN2/MAINT,             ;IS THAT YOU, VICKIE?
                  ;13846 ;       NAD/DRIVE.TYPE
                  ;13847 ;
                  ;13848 ;-----------------------------------------------------------------
                  ;13849 ;=0**
                  ;13850 ;
                  ;13851 ;DRIVE.TYPE:
                  ;13852 ;
                  ;13853 ;       BEN0/R80,               ;WHICH TYPE OF DRIVE WAS SELECTED?
                  ;13854 ;       NAD/RL02.IFORK
                  ;13855 ;
                  ;13856 ;-----------------------------------------------------------------
                  ;13857 ;=1
                  ;13858 ;
                  ;13859 ;R80.IFORK:
                  ;13860 ;
                  ;13861 ;       UCMD/SET.FIFOA,
                  ;13862 ;       FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
                  ;13863 ;       BEN0/F0,                ;8 WAY INSTRUCTION FORK.
                  ;13864 ;       BEN1/F1,                ;BRANCH ON F<2:0> IN THE CSR.
                  ;13865 ;       BEN2/F2,
;13866 ;        NAD/#NOP.CMD
;13867 ;
;13868 ;--------------------------------------------------------------------------
;13869 ;=110
;13870 ;
;13871 ;#RD.DAT.CMD:
;13872 ;
;13873 ;        SET.FIFO.ADR,           ;START OF THE R80 READ DATA COMMAND
;13874 ;        NAD/#INIT.FIFO6         ;PRESET FIFOA ADRS CNTR TO OVFLW.
;13875 ;
;13876 ;--------------------------------------------------------------------------
;13877 ;
;13878 ;#INIT.FIFO6:
;13879 ;
;13880 ;        UCMD/CHG.FIFO           ;CHANGE TO FIFOB
;13881 ;
;13882 ;--------------------------------------------------------------------------
;13883 ;
;13884 ;#INIT.FIFO6A:
;13885 ;
;13886 ;        SET.FIFO.ADR            ;PRESET FIFOB ADRS CNTR TO OVFLW.
;13887 ;
;13888 ;--------------------------------------------------------------------------
;13889 ;
;13890 ;#INIT.FIFO6B:
;13891 ;
;13892 ;        UCMD/SET.FIFOA,         ;RETURN TO FIFOA
;13893 ;        NAD/#WAIT.DRDY1
;13894 ;
;13895 ;--------------------------------------------------------------------------
;13896 ;=0
;13897 ;
;13898 ;#WAIT.DRDY1:
;13899 ;
;13900 ;        BEN0/DRIVE.RDY,         ;WAIT FOR DRIVE READY
;13901 ;        NAD/#WAIT.DRDY1
;13902 ;
;13903 ;--------------------------------------------------------------------------
;13904 ;=1
;13905 ;
;13906 ;#CHGCLK1:
;13907 ;
;13908 ;        ENB.SYS.CLK,            ;USE R80 SYSTEM CLOCK
;13909 ;        NAD/#WAIT.NOSEC.PLS1
;13910 ;
;13911 ;--------------------------------------------------------------------------
;13912 ;=1*
;13913 ;
;13914 ;#WAIT.NOSEC.PLS1:
;13915 ;
;13916 ;        BEN1/SECTOR.PLS,        ;PREPARE TO LOOK FOR THE LEADING EDGE
;13917 ;        NAD/#WAIT.NOSEC.PLS1    ;OF SECTOR PULSE.
;13918 ;
;13919 ;--------------------------------------------------------------------------
;13920 ;=0*
```

```
;13921 ;
;13922 ;#NOSEC.PLS1:
;13923 ;
;13924 ;        NAD/#WAIT.SEC.PLS1          ;WE ARE NOT IN SECTOR PULSE AREA.
;13925 ;
;13926 ;--------------------------------------------------------------------------------
;13927 ;=0*
;13928 ;
;13929 ;#WAIT.SEC.PLS1:
;13930 ;
;13931 ;        BEN1/SECTOR.PLS,            ;WAIT FOR LEADING EDGE OF SEC PLS
;13932 ;        NAD/#WAIT.SEC.PLS1
;13933 ;
;13934 ;--------------------------------------------------------------------------------
;13935 ;=1*
;13936 ;
;13937 ;#SEC.PLS.FND1:
;13938 ;
;13939 ;        LOAD.LOOP.CNTR.[0C5],       ;SECTOR PULSE HAS BEEN FOUND.
;13940 ;        NAD/#WAIT.60.TICKS1
;13941 ;
;13942 ;--------------------------------------------------------------------------------
;13943 ;=0
;13944 ;
;13945 ;#WAIT.60.TICKS1:
;13946 ;
;13947 ;        INCR.LPCNTR/ON,             ;DELAY BEFORE ASSERTING READ GATE
;13948 ;        TAG.STROBE/ON,
;13949 ;        BEN0/CNTR.OVFLW,
;13950 ;        NAD/#WAIT.60.TICKS1
;13951 ;
;13952 ;--------------------------------------------------------------------------------
;13953 ;=1
;13954 ;
;13955 ;#SETUP.LOOP.LOCK1:
;13956 ;
;13957 ;        LOOP.LOCK/ON,               ;ALLOW PHASE LOCK LOOP TO SYNC ON
;13958 ;        TAG.STROBE/ON,              ;ALL ZEROS PATTERN
;13959 ;        FUNC/CLR.ECC,
;13960 ;        LOAD.LOOP.CNTR.[0A9]
;13961 ;
;13962 ;--------------------------------------------------------------------------------
;13963 ;=0
;13964 ;
;13965 ;#WAIT.LOOP.LOCK1:
;13966 ;
;13967 ;        LOOP.LOCK/ON,               ;WAIT FOR 88 TICKS AFTER READ GATE
;13968 ;        TAG.STROBE/ON,
;13969 ;        UCON/CLR.MATCH,
;13970 ;        INCR.LPCNTR/ON,
;13971 ;        BEN0/CNTR.OVFLW,
;13972 ;        NAD/#WAIT.LOOP.LOCK1
;13973 ;
;13974 ;--------------------------------------------------------------------------------
;13975 ;=1
```

```
;13976 ;
;13977 ;#SETUP.RD.HDR1:
;13978 ;
;13979 ;        LOOP.LOCK/ON,               ;SETUP INITIAL CONDITIONS
;13980 ;        TAG.STROBE/ON,              ;FOR READING HEADER.
;13981 ;        FUNC/CLR.DSR,
;13982 ;        ENB.CLR.BC/ON,
;13983 ;        LOAD.LOOP.CNTR.[0E1]
;13984 ;
;13985 ;----------------------------------------------------------------
;13986 ;
;13987 ;#CHGCLK1A:
;13988 ;
;13989 ;        ENB.DSK.CLK,                ;THE MICROSEQUENCER WILL STALL
;13990 ;        LOOP.LOCK/ON,               ;UNTIL SYNC SEEN IS SET
;13991 ;        TAG.STROBE/ON,              ;(SYNC BYTE FOUND)
;13992 ;        ENB.CLR.BC/ON,
;13993 ;        GEN.SYNC.[19]
;13994 ;
;13995 ;----------------------------------------------------------------
;13996 ;=0
;13997 ;
;13998 ;#READ.HDR1:
;13999 ;
;14000 ;        LOOP.LOCK/ON,               ;SHIFT IN 32 HEADER BITS AND
;14001 ;        TAG.STROBE/ON,              ;COMPARE WITH SERIAL DAR
;14002 ;        DATA.FIELD/ON,
;14003 ;        ENB.CLR.BC/ON,
;14004 ;        INCR.LPCNTR/ON,
;14005 ;        BEN0/CNTR.OVFLW,
;14006 ;        NAD/#READ.HDR1
;14007 ;
;14008 ;----------------------------------------------------------------
;14009 ;=1
;14010 ;
;14011 ;#TEST.HDR.MATCH1:
;14012 ;
;14013 ;        UCON/TEST.MATCH,
;14014 ;        LOOP.LOCK/ON,               ;HEADER HAS BEEN READ.
;14015 ;        TAG.STROBE/ON,              ;TURN ON TEST MATCH, OFF ENB CLR BC.
;14016 ;        DATA.FIELD/ON,              ;PREPARE TO READ THE CRC WORD.
;14017 ;        NAD/#BRANCH.HDR.MATCH1
;14018 ;
;14019 ;----------------------------------------------------------------
;14020 ;
;14021 ;#BRANCH.HDR.MATCH1:
;14022 ;
;14023 ;        LOOP.LOCK/ON,
;14024 ;        TAG.STROBE/ON,
;14025 ;        CHECK.FIELD/ON,             ;RESULTS OF HDR COMP NOW IN MISMATCH.
;14026 ;        BEN2/MISMATCH.L,
;14027 ;        NAD/#HDR.COMP1
;14028 ;
;14029 ;----------------------------------------------------------------
;14030 ;=1**
```

```
;
;@HDR.COMP1:
;
;        LOOP.LOCK/ON,               ;HEADER COMPARE
;        TAG.STROBE/ON,
;        CHECK.FIELD/ON,             ;CONTINUE TO READ THE CRC WORD.
;        LOAD.LOOP.CNTR.[OF3],
;        BENO/CSSE,                  ;CHECK FOR SKIP SECTOR ERROR.
;        NAD/@NOT.SSE1
;
;-----------------------------------------------------------------------
;=0
;
;@NOT.SSE1:
;
;        LOOP.LOCK/ON,
;        TAG.STROBE/ON,
;        CHECK.FIELD/ON,             ;CONTINUE TO READ THE CRC WORD.
;        NAD/@RD.CRC.WD1
;
;-----------------------------------------------------------------------
;=0
;
;@RD.CRC.WD1:
;
;        LOOP.LOCK/ON,               ;CONTINUE TO SHIFT IN THE 16 BIT
;        TAG.STROBE/ON,              ;CRC WORD.
;        CHECK.FIELD/ON,
;        INCR.LPCNTR/ON,
;        BENO/CNTR.OVFLW,
;        NAD/@RD.CRC.WD1
;
;-----------------------------------------------------------------------
;=1
;
;@CHECK.CRC1:
;
;        LOOP.LOCK/ON,               ;DROP CFLD. CLOCK ECC (CRC) FLOP.
;        TAG.STROBE/ON,
;        UCON/CHECK.ECC
;
;-----------------------------------------------------------------------
; This is a new state for U14IDC to allow DC007 time to
; settle error bits.
;
;@CHECK.CRC2:
;
;        LOOP.LOCK/ON,               ;DROP CFLD. CLOCK ECC (CRC) FLOP.
;        TAG.STROBE/ON,
;        UCON/CHECK.ECC
;
;-----------------------------------------------------------------------
;
;@TEST.FOR.ERR1:
;
```

```
;14086 ;        LOOP.LOCK/ON,           ;WAS THERE A CRC ERROR?
;14087 ;        TAG.STROBE/ON,
;14088 ;        BEN1/ERROR.L
;14089 ;
;14090 ;-----------------------------------------------------------------
;14091 ;=1*
;14092 ;
;14093 ;#NOT.CRC.ERR1:
;14094 ;
;14095 ;        LOOP.LOCK/ON,           ;CLEAR DC007
;14096 ;        TAG.STROBE/ON,
;14097 ;        FUNC/CLR.ECC,
;14098 ;        BEN2/F2,                ;BRANCH ON FUNCTION BITS INTO WRT CHK,
;14099 ;        BEN1/F1,                ;RD DATA OR WRT DATA FLOWS.
;14100 ;        NAD/#BRANCH.WC.CMD1
;14101 ;
;14102 ;-----------------------------------------------------------------
;14103 ;=11*
;14104 ;
;14105 ;#BRANCH.RD.CMD6:
;14106 ;
;14107 ;        LOOP.LOCK/ON,           ;CONTINUE VIA READ DATA FLOWS.
;14108 ;        TAG.STROBE/ON,
;14109 ;        BEN1/FIFO.OVFLW.L,
;14110 ;        NAD/#CHGCLK6B
;14111 ;
;14112 ;-----------------------------------------------------------------
;14113 ;=0*
;14114 ;
;14115 ;#CHGCLK6B:
;14116 ;
;14117 ;        LOOP.LOCK/ON,           ;EVERYTHING IS GO FOR READING DATA
;14118 ;        TAG.STROBE/ON,          ;FROM THIS SECTOR. RUN OFF SYSTEM CLOCK
;14119 ;        ENB.SYS.CLK,            ;THRU HEADER POSTAMBLE AND PAST SPLICE.
;14120 ;        LOAD.LOOP.CNTR.[0F6],
;14121 ;        NAD/#HDR.POST6
;14122 ;
;14123 ;-----------------------------------------------------------------
;14124 ;=0
;14125 ;
;14126 ;#HDR.POST6:
;14127 ;
;14128 ;        TAG.STROBE/ON,          ;DROP READ GATE.
;14129 ;        INCR.LPCNTR/ON,         ;SHIFT OVER WRITE SPLICE.
;14130 ;        BEN0/CNTR.OVFLW,
;14131 ;        NAD/#HDR.POST6
;14132 ;
;14133 ;-----------------------------------------------------------------
;14134 ;=1
;14135 ;
;14136 ;#DATA.PRE6:
;14137 ;
;14138 ;        LOOP.LOCK/ON,           ;TURN ON READ GATE AGAIN
;14139 ;        TAG.STROBE/ON,          ;NOW THAT WE'RE IN DATA PREAMBLE.
;14140 ;        FUNC/CLR.ECC,
;14141 ;        LOAD.LOOP.CNTR.[0A9]
;14142 ;
;14143 ;---------------------------------------------------------------------------
;14144 ;=0
;14145 ;
;14146 ;#DATA.PRE6A:
;14147 ;
;14148 ;        LOOP.LOCK/ON,               ;ALLOW PHASE LOCK LOOP 11 BYTES
;14149 ;        TAG.STROBE/ON,              ;OF PREAMBLE TO LOCK UP.
;14150 ;        INCR.LPCNTR/ON,
;14151 ;        BENO/CNTR.OVFLW,
;14152 ;        NAD/#DATA.PRE6A
;14153 ;
;14154 ;---------------------------------------------------------------------------
;14155 ;=1
;14156 ;
;14157 ;#SETUP.RD.DATA6:
;14158 ;
;14159 ;        LOOP.LOCK/ON,               ;SETUP INITIAL CONDITIONS
;14160 ;        TAG.STROBE/ON,              ;FOR READING DATA.
;14161 ;        CLR.FIFO.ADR,
;14162 ;        LOAD.LOOP.CNTR.[0F9]
;14163 ;
;14164 ;---------------------------------------------------------------------------
;14165 ;
;14166 ;#CHGCLK6C:
;14167 ;
;14168 ;        LOOP.LOCK/ON,               ;THE MICROSEQUENCER WILL STALL UNTIL
;14169 ;        TAG.STROBE/ON,              ;SYNC SEEN IS SET (SYNC BYTE FOUND).
;14170 ;        MODE/ECC,
;14171 ;        ENB.DSK.CLK,                ;THEN WE USE READ CLOCK FROM R80.
;14172 ;        GEN.SYNC.[19],
;14173 ;        NAD/#READ.DATA6
;14174 ;
;14175 ;---------------------------------------------------------------------------
;14176 ;=10
;14177 ;
;14178 ;#READ.DATA6:
;14179 ;
;14180 ;        LOOP.LOCK/ON,               ;SHIFT IN 7 DATA BITS.
;14181 ;        TAG.STROBE/ON,
;14182 ;        DATA.FIELD/ON,
;14183 ;        MODE/ECC,
;14184 ;        INCR.LPCNTR/ON,
;14185 ;        BENO/CNTR.OVFLW,
;14186 ;        NAD/#READ.DATA6
;14187 ;
;14188 ;---------------------------------------------------------------------------
;14189 ;=11
;14190 ;
;14191 ;#STORE.BYTE6:
;14192 ;
;14193 ;        LOOP.LOCK/ON,               ;8TH DATA BIT
;14194 ;        TAG.STROBE/ON,
;14195 ;        DATA.FIELD/ON,
;14196 ;         MODE/ECC,
;14197 ;         ENB.DSR/ON,
;14198 ;         WRT.FIFO/ON,             ;LOAD BYTE INTO FIFO
;14199 ;         LOAD.LOOP.CNTR.[0F9],
;14200 ;         INCR.FIFO.CNTR/ON,
;14201 ;         BEN1/FIFO.MAX.L,
;14202 ;         NAD/#READ.DATA6
;14203 ;-----------------------------------------------------------------------
;14204 ;=00
;14205 ;
;14206 ;#RD.DATA.ECC6:
;14207 ;
;14208 ;         LOOP.LOCK/ON,            ;START TO READ THE ECC WORD
;14209 ;         TAG.STROBE/ON,
;14210 ;         CHECK.FIELD/ON,          ;DROP DFLD, ASSERT CFLD.
;14211 ;         MODE/ECC,
;14212 ;         LOAD.LOOP.CNTR.[0E1],
;14213 ;         NAD/#RD.DATA.ECC6A
;14214 ;
;14215 ;-----------------------------------------------------------------------
;14216 ;=0
;14217 ;
;14218 ;#RD.DATA.ECC6A:
;14219 ;
;14220 ;         LOOP.LOCK/ON,            ;SHIFT IN REST OF 32 BIT FIRE CODE
;14221 ;         TAG.STROBE/ON,
;14222 ;         CHECK.FIELD/ON,
;14223 ;         MODE/ECC,
;14224 ;         INCR.LPCNTR/ON,
;14225 ;         BEN0/CNTR.OVFLW,
;14226 ;         NAD/#RD.DATA.ECC6A
;14227 ;
;14228 ;-----------------------------------------------------------------------
;14229 ;=1
;14230 ;
;14231 ;#CHECK.ECC6:
;14232 ;
;14233 ;         LOOP.LOCK/ON,            ;DROP CFLD. CLOCK ECC FLOP.
;14234 ;         TAG.STROBE/ON,
;14235 ;         MODE/ECC,
;14236 ;         UCON/CHECK.ECC
;14237 ;
;14238 ;-----------------------------------------------------------------------
;14239 ; Extra state inserted in U14IDC to allow DC007 time to settle
;14240 ;
;14241 ;#CHECK.ECC6A:
;14242 ;
;14243 ;         LOOP.LOCK/ON,            ;DROP CFLD. CLOCK ECC FLOP.
;14244 ;         TAG.STROBE/ON,
;14245 ;         MODE/ECC,
;14246 ;         UCON/CHECK.ECC
;14247 ;
;14248 ;-----------------------------------------------------------------------
;14249 ;
;14250 ;#TEST.FOR.ERR6A:
```

```
;
;       LOOP.LOCK/ON,
;       TAG.STROBE/ON,
;       MODE/ECC,
;       CLR.FIFO.ADR,
;       BEN1/ERROR.L               ;WAS THERE AN ECC ERROR?
;
;-----------------------------------------------------------------------
;=1*
;
;#NOT.ECC.ERR6:
;
;       LOOP.LOCK/ON,
;       TAG.STROBE/ON,             ;SWITCH OVER TO CPU CLOCK
;       ENB.CPU.CLK                ;FOR FAST INTR OR TERMINATE.
;
;-----------------------------------------------------------------------
;
;#XFER.RQST6:
;
;       UCMD/SET.XFER.RQST,        ;SEND FAST INTERRUPT TO CPU.
;       NAD/#WAIT.XFER.GRANT6
;
;-----------------------------------------------------------------------
;=1*
;
;#WAIT.XFER.GRANT6:
;
;       BEN1/XFER.REQ,             ;CPU WILL TURN ON CRDY IF IT'S THE LAST
;       NAD/#WAIT.XFER.GRANT6      ;SECTOR, THEN SEND GRANT.
;
;-----------------------------------------------------------------------
;=0*
;
;#TEST.XFER.DONE6:
;
;       UCMD/CHG.FIFO,             ;SWITCH FIFO'S, IN CASE THERE IS MORE
;       BEN0/CRDY.L,               ;DATA TO BE READ.
;       NAD/#XFER.DONE6
;
;-----------------------------------------------------------------------
;=0
;
;#XFER.DONE6:
;
;       UCMD/SET.CRDY,
;       INCR.DAR/ON,
;       FUNC/SET.INT,              ;CPU SIGNALS END OF TRANSFER.
;       NAD/SET.INT.AND.IDLE
;
;-----------------------------------------------------------------------
;=111111110
;
;SET.INT.AND.IDLE:
;
```

```
;14306 ;        FUNC/SET.INT,           ; The purpose of this state is to give CRDY one
;14307 ;        NAD/IDLE                ; state to reset before asserting interrupt.
;14308 ;                                ; CRDY is now used in the INT equation.
;14309 ;
;14310 ;-----------------------------------------------------------------------
;14311 ;=1
;14312 ;
;14313 ;RTN.TO.IDLE:
;14314 ;IDLE:
;14315 ;
;14316 ;        BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
;14317 ;        NAD/SETUP.DRV.CHK
;14318 ;
;14319 ;-----------------------------------------------------------------------
;14320 ;=0
;14321 ;
;14322 ;SETUP.DRV.CHK:
;14323 ;
;14324 ;        LOAD.LOOP.CNTR.[OFB],   ;COUNTER WILL BE USED
;14325 ;        NAD/SEL.NXT.DRV         ;TO SEQUENCE THRU DRIVES.
;14326 ;
;14327 ;-----------------------------------------------------------------------
;14328 ;
;14329 ;SEL.NXT.DRV:
;14330 ;
;14331 ;        SEL.DRV/UPROG,
;14332 ;        BENO/CNTR.OVFLW,
;14333 ;        NAD/CHK.DRV.STATUS
;14334 ;
;14335 ;-----------------------------------------------------------------------
;14336 ;=0
;14337 ;
;14338 ;CHK.DRV.STATUS:
;14339 ;        ; Give drive status time to settle down.
;14340 ;
;14341 ;        SEL.DRV/UPROG,          ;SELECT DRIVE
;14342 ;        NAD/DRV.STATUS.DISPATCH
;14343 ;
;14344 ;-----------------------------------------------------------------------
;14345 ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
;14346 ;
;14347 ;DRV.STATUS.DISPATCH:
;14348 ;        SEL.DRV/UPROG,          ;SELECT DRIVE
;14349 ;        BEN1/DRIVE.ONLINE,      ;AND BRANCH ON STATUS.
;14350 ;        BENO/DRIVE.RDY,
;14351 ;        NAD/-ERR.-OL.-RDY
;14352 ;
;14353 ;
;14354 ;-----------------------------------------------------------------------
;14355 ;=011
;14356 ;
;14357 ;-ERR.OL.RDY:
;14358 ;
;14359 ;        SEL.DRV/UPROG,          ;NOP
;14360 ;        INCR.LPCNTR/ON,         ;ADD 1 TO DRIVE SELECT.
```

K3

```
                   ;14361 ;        NAD/SEL.NXT.DRV
                   ;14362 ;
                   ;14363 ;-------------------------------------------------------------
                   ;14364 ;
                   ;14365 ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                   ;14366 ;
                   ;14367 ;                UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                   ;14368 ;
                   ;14369 ;-------------------------------------------------------------
                   ;14370 ;
                   ;14371 ;=1
                   ;14372 ;
                   ;14373 ;RTN.TO.IDLE:
                   ;14374 ;IDLE:
                   ;14375 ;
                   ;14376 ;        BENO/CRDY.L,              ;LOOK FOR GO (CRDY L)
                   ;14377 ;        NAD/SETUP.DRV.CHK
                   ;14378 ;
                   ;14379 ;-------------------------------------------------------------
                   ;14380
                   ;14381 ;Set up parameters for timeout error report :
                   ;14382
U 0BD0, B6,4295    ;14383         MOV LS[#2] TO WR[1]
U 0BD1, 3E,8295    ;14384         MOV WR[1] TO LS[ERROR.NUMBER]             ;set up error number = 2 in LS
U 0BD2, 36,4A95    ;14385         MOV LS[IDC] TO WR[1]                      ;store module code in LS to
U 0BD3, BE,8C95    ;14386         MOV WR[1] TO LS[MODULE.NUM]               ; modules
U 0BD4, 36,6E95    ;14387         MOV LS[NA.EXP.REC] TO WR[1]               ;put "N/A" for printout data
U 0BD5, BE,8095    ;14388         MOV WR[1] TO LS[CONTROL.STATUS]
                   ;14389
U 0BD6, 2F,8015    ;14390         CLR WR[0]                                 ;Initialize time-out counter
U 0BD7, BF,8C15    ;14391         MOV WR[0] TO LS[TIMEOUT.CNTR]
                   ;14392
                   ;14393 WAIT.IDC.TD.2:
U 0BD8, 37,8C15    ;14394         MOV LS[TIMEOUT.CNTR] TO WR[0]             ;load the current time count
U 0BD9, 37,9095    ;14395         MOV LS[WAIT.TIME1] TO WR[1]               ;time limit on read data
                   ;14396         CMP WR[0] WITH WR[1],                     ;timeout yet ???
U 0BDA, 26,4085    ;14397          DT(LONG)&SET.ALU.CC
U 0BDB, 88,BE19    ;14398         JMP.IF[EQL] TO [TD.ERR.2]                 ;yes, timeout on read data
U 0BDC, 20,4015    ;14399         INC WR[0]                                 ;inc. time count
U 0BDD, BF,8C15    ;14400         MOV WR[0] TO LS[TIMEOUT.CNTR]
U 0BDE, DB,001A    ;14401         SKIP.IF[NO.FAST.INTERRUPT]                ;is XFER REQ set ???
U 0BDF, 88,BE64    ;14402         JMP [IDC.DONE.TD.2]                       ;yes, XFER REQ is set now
U 0BE0, 08,BD84    ;14403         JMP [WAIT.IDC.TD.2]                       ;no, wait some more
                   ;14404
                   ;14405 ;Force an error in the CHECK.RESULT routine for this timeout :
                   ;14406
                   ;14407 TD.ERR.2:
U 0BE1, 2F,8015    ;14408         CLR WR[0]                                 ;set unequal values
U 0BE2, 36,4095    ;14409         MOV LS[#1] TO WR[1]                       ; between WR0 and WR1
U 0BE3, 08,693C    ;14410         JSR [CHECK.RESULT]                        ;force an error
U 0BE4, 88,BC84    ;14411         JMP [T.D.7]                               ;err and if LOE is set
U 0BE5, 88,BE64    ;14412         JMP [IDC.DONE.TD.2]
                   ;14413
                   ;14414 IDC.DONE.TD.2:
U 0BE6, 90,FA15    ;14415         MISC2 [TRANSFER.GRANT]                    ;clear the XFER REQ
```

```
                    ;14416
                    ;14417 ;Set up error parameters for ERROR 3 report :
                    ;14418
U 0BE7, 36,C015     ;14419          MOV LS[#3] TO WR[0]                     ;error number = 3
U 0BE8, BE,8215     ;14420          MOV WR[0] TO LS[ERROR.NUMBER]
U 0BE9, B6,2C15     ;14421          MOV LS[#FFFFFF00] TO WR[0]              ;error mask for R80
U 0BEA, 3E,8A15     ;14422          MOV WR[0] TO LS[ERROR.MASK]             ; expected status
U 0BEB, 36,4A95     ;14423          MOV LS[IDC] TO WR[1]                    ;store IDC and R80 in LS
U 0BEC, C7,4E95     ;14424          BIS LS[R80] TO WR[1]                    ; as module codes to indicate
U 0BED, BE,8C95     ;14425          MOV WR[1] TO LS[MODULE.NUM]             ;  modules at fault
U 0BEE, E5,8015     ;14426          CLR LS[CONTROL.STATUS]                  ;clr [NA.EXP.REC]
                    ;14427
                    ;14428 ;Read the data from FIFO A and see if the data received is the same as expected :
                    ;14429
U 0BEF, 17,D815     ;14430          PORT.CONTROL [SEL.FIFO.A]               ;read FIFO A
U 0BF0, 17,C015     ;14431          PORT.CONTROL [CLEAR.FIFO.CNTR]          ;clear FIFO A adr.
U 0BF1, 90,9015     ;14432          MISC [SET.PORT.I.O]                     ;select IDC
U 0BF2, 17,8815     ;14433          PORT.READ PORT.ADRS[DATA.BYTE]          ;read a data byte
U 0BF3, 3B,3215     ;14434          MOV PORT TO LS[PORT0]                   ; from the FIFO into
U 0BF4, 37,3215     ;14435          MOV LS[PORT0] TO WR[0]                  ;  WR0 as the received data
U 0BF5, 36,9895     ;14436          MOV LS[#AAAAAAAA] TO WR[1]              ;expected data
U 0BF6, 08,693C     ;14437          JSR [CHECK.RESULT]                      ;data read matches expected
U 0BF7, 88,BC84     ;14438          JMP [T.D.7]                             ;err and if LOE is set
U 0BF8, 08,C044     ;14439          JMP [END.TD]                            ;test ends
                    ;14440
                    ;14441 BAD.SECTOR.TD.1:
U 0BF9, B6,4495     ;14442          MOV LS[#4] TO WR[1]
U 0BFA, 3E,8295     ;14443          MOV WR[1] TO LS[ERROR.NUMBER]           ;set up error number = 4 in LS
U 0BFB, E5,8A15     ;14444          CLR LS[ERROR.MASK]                      ;no error mask used here
U 0BFC, B6,4E95     ;14445          MOV LS[R80] TO WR[1]                    ;store module code in LS to
U 0BFD, BE,8C95     ;14446          MOV WR[1] TO LS[MODULE.NUM]             ; modules
U 0BFE, 36,6E95     ;14447          MOV LS[NA.EXP.REC] TO WR[1]             ;put "N/A" for printout data
U 0BFF, BE,8095     ;14448          MOV WR[1] TO LS[CONTROL.STATUS]
                    ;14449
                    ;14450 ;Force an error to indicate that there are no good sector found on this track :
                    ;14451
U 0C00, 2F,8015     ;14452          CLR WR[0]                               ;set received data and
U 0C01, 36,4095     ;14453          MOV LS[#1] TO WR[1]                     ; expected data unequal
U 0C02, 08,693C     ;14454          JSR [CHECK.RESULT]                      ;forcing the error
U 0C03, 88,BC84     ;14455          JMP [T.D.7]                             ;err and if LOE is set
                    ;14456
                    ;14457 END.TD:                                          ;end of test
                    ;14458
```

```
;14459 .PAGE  "Test E  R80 WRITE CHECK FUNCTION TEST (not run in CRD)"
;14460 ;++
;14461 ;
;14462 ; Test Description:
;14463 ;
;14464 ;        This test will verify the R80 WRITE CHECK function by first writing
;14465 ;        data to the disk similar to the operation procedure as in the "R80
;14466 ;        WRITE DATA FUNCTION TEST" described above. And then the test will
;14467 ;        load the data buffer (FIFO A) with the data just used in the write
;14468 ;        data function, and assert the CSR <bit 1>. The test will wait for a
;14469 ;        PORT XFER REQ signal from the IDC microcode as a sign of no write
;14470 ;        check error.
;14471 ;
;14472 ; Assumption:
;14473 ;
;14474 ;        Both CPU micros and IDC part I has been run successfully and
;14475 ;        drive 0 is a R80, up and spinning and ready to accept commands;
;14476 ;        R80 get status function is working; R80 recalibrate, initialization,
;14477 ;        seek and read header, write data and read data functions are working
;14478 ;        properly.
;14479 ;
;14480 ;        This test assumes "DISK DRIVE SIZING TEST" has been run prior to this
;14481 ;        test.
;14482 ;
;14483 ; Test steps:
;14484 ;
;14485 ;         1. Load LS[DR0.R80] and examine its content
;14486 ;         2. If value is zero, go to end of test
;14487 ;         3. Else seek to track 561, wait for CRDY & DRDY bits of CSR to set
;14488 ;         4. Init. the sector to be written on
;14489 ;         5. Load FIFO A with a data pattern
;14490 ;         6. Load DAR with the selected sector, cylinder 561, head 3
;14491 ;         7. Invoke the write data function to the R80; CSR <3 & 1>
;14492 ;         8. Wait for XFER REQ to set
;14493 ;         9. Check if the current sector is a good sector, if bad
;14494 ;        10.     Then see if it has tried out all 31 sector on this track, if so;
;14495 ;        11.             Then report ERROR 2
;14496 ;        12.             Else inc. to the next higher sector
;14497 ;        13.                     and go to step 5
;14498 ;        14.     Else load FIFO A the same data
;14499 ;        15. Load DAR with the selected sector and cylinder 561, head 3
;14500 ;        16. Invoke the write check function to the R80; CSR <1>
;14501 ;        17. Set up a time out loop waiting for XFER REQ to set
;14502 ;        18. Set up pertinent error parameters for ERROR 1
;14503 ;        19. If time out, report the error
;14504 ;        20. Else test ends
;14505 ;
;14506 ; Error:
;14507 ;
;14508 ; ERROR 1 - Time out waiting for XFER REQ to set after a Write Check
;14509 ;           command has been issued to the IDC
;14510 ;
;14511 ;          Printout:
;14512 ;               EXPECTED        RECEIVED        MODULE
;14513 ;               N/A             N/A             IDC
```

```
                    ;14514 ;
                    ;14515 ; Debug:
                    ;14516 ;
                    ;14517 ;   ERROR 1: Check DRIVE READY, R80 SYSTEM CLOCK, SYNC SEEN, DS CLOCK,
                    ;14518 ;            R80 READ DATA & WRT CHK LOAD for fault. Check PAL 16R4
                    ;14519 ;            (DSK RD SEQ REG) and its signals: DSR 0, COMPARE,
                    ;14520 ;            UWRT CHK, WRTCHK LOAD DSR.
                    ;14521 ;
                    ;14522 ;--
                    ;14523
                    ;14524 T.E:
U 0C04, B6,5E15     ;14525         MOV LS[BEGIN.TEST] TO WR[0]             ;set bit 15 in WR[0] for
                    ;14526                                                 ; control and status word
U 0C05, 3E,8015     ;14527         MOV WR[0] TO LS[CONTROL.STATUS]         ;set bit 15 in control and
                    ;14528                                                 ; status word - bit 15
                    ;14529                                                 ; indicates start of test to
                    ;14530                                                 ; console
U 0C06, 10,E015     ;14531         MISC [SET.CP.ATTN]                      ;issue CPU ATTN to console
                    ;14532
                    ;14533 WAIT.TE.0:
U 0C07, 08,C074     ;14534         JMP [WAIT.TE.0]                         ;loop here waiting for
                    ;14535                                                 ; console to respond
                    ;14536
U 0C08, 65,A015     ;14537         CLR LS[PREVIOUS.ERROR]                  ;clear previous error number
                    ;14538                                                 ; in LS
U 0C09, 17,CC15     ;14539         PORT.CONTROL [CLEAR.IDC]                ;init. IDC
U 0C0A, 17,CC15     ;14540         PORT.CONTROL [CLEAR.IDC]
U 0C0B, 17,CC15     ;14541         PORT.CONTROL [CLEAR.IDC]
                    ;14542
                    ;14543 T.E.1:
U 0C0C, B7,0C15     ;14544         MOV LS[DR0.R80] TO WR[0]                ;load the R80 presence indicator
                    ;14545         TST WR[0],                              ;see if drive one has an R80
U 0C0D, 20,0035     ;14546          DT(LONG)&SET.ALU.CC
U 0C0E, 08,C7E9     ;14547         JMP.IF[EQL] TO [END.TE]                 ;no R80, go to end of test
                    ;14548
                    ;14549 ;In order to do a seek on the R80, a head select operation must be first initiated
                    ;14550 ; and it is then followed by the actual seek :
                    ;14551
                    ;14552 HEAD.SELECT.TE.1:
U 0C0F, 36,4215     ;14553         MOV LS[#2] TO WR[0]                     ;selecting the third head
U 0C10, 47,5C15     ;14554         BIS LS[BIT14] TO WR[0]                  ;set head select code
U 0C11, 97,A415     ;14555         PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ;load IDC DAR
                    ;14556
U 0C12, 37,0415     ;14557         MOV LS[SEEK] TO WR[0]                   ;fetch the seek command
U 0C13, 47,9C15     ;14558         BIS LS[SETCSR.DS0] TO WR[0]             ;select drive zero
U 0C14, 17,A015     ;14559         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;initiate the head select command
                    ;14560
                    ;14561 ;It is necessary to wait for CRDY before issuing another command or destroying the
                    ;14562 ; DAR :
                    ;14563
                    ;14564 WAIT.HEAD.SELECT.DONE.TE.1:
U 0C15, 88,700C     ;14565         JSR [WAIT.CSR.CRDY]                     ;check if CRDY is ready
U 0C16, 88,C184     ;14566         JMP [CLR.ATN.BITS.TE.1]                 ;yes, CRDY is set now
U 0C17, 08,C154     ;14567         JMP [WAIT.HEAD.SELECT.DONE.TE.1]        ;no, wait some more
                    ;14568
```

```
                   ;14569 ;This section of code is to insure that the proper attention bit is cleared after
                   ;14570 ; the implied head select :
                   ;14571
                   ;14572 CLR.ATN.BITS.TE.1:
U 0C18, 36,0A15    ;14573         MOV LS[T5.SUB] TO WR[0]                  ;bring in the CSR word
U 0C19, B6,6695    ;14574         MOV LS[BIT19] TO WR[1]                   ;CSR ATN3 bit
U 0C1A, 47,6495    ;14575         BIS LS[BIT18] TO WR[1]                   ;add CSR ATN2 bit
U 0C1B, AF,4215    ;14576         BIC WR[1] TO WR[0]                       ;reset ATN3:ATN2 bits
U 0C1C, 17,A015    ;14577         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]     ;clear attn. bits in CSR
                   ;14578
                   ;14579 ;Start the actual seek to the cylinder 561 :
                   ;14580
                   ;14581 ACTUAL.R80.SEEK.TE:
U 0C1D, 37,B615    ;14582         MOV LS[#230] TO WR[0]                    ;cylinder address for DAR format
U 0C1E, 47,5A15    ;14583         BIS LS[BIT13] TO WR[0]                   ;specify this is a cyl. adr. word
U 0C1F, 97,C415    ;14584         PORT.CONTROL [RESET.BR]                  ;clear interrupt caused by head select
U 0C20, 97,A415    ;14585         PORT.WRITE PORT.ADRS[DAR] WITH WR[0]     ;write to the DAR
                   ;14586
U 0C21, 37,0415    ;14587         MOV LS[SEEK] TO WR[0]                    ;fetch the seek command
U 0C22, 47,9C15    ;14588         BIS LS[SETCSR.DS0] TO WR[0]              ;select drive zero
U 0C23, 17,A015    ;14589         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]     ;start the R80 seek
                   ;14590
                   ;14591     ;*************************************************************************
                   ;14592     ;* REFER TO THE R80 SEEK IDC FUNCTION CODES AS LISTED IN THIS TEST ABOVE *
                   ;14593     ;*************************************************************************
                   ;14594
                   ;14595 WAIT.IDC.SEEK.TE.1:
U 0C24, 88,700C    ;14596         JSR [WAIT.CSR.CRDY]                      ;check if CRDY is set
U 0C25, 88,C274    ;14597         JMP [CHECK.DRDY.TE.1]                    ;yes, next test
U 0C26, 88,C244    ;14598         JMP [WAIT.IDC.SEEK.TE.1]                 ;wait some more
                   ;14599
                   ;14600 ;Check if the R80 is ready to take the next operation :
                   ;14601
                   ;14602 CHECK.DRDY.TE.1:
U 0C27, 08,708C    ;14603         JSR [WAIT.CSR.DRDY]                      ;wait for DRDY to set
U 0C28, 36,9C15    ;14604         MOV LS[ZERO] TO WR[0]
U 0C29, BF,8015    ;14605         MOV WR[0] TO LS[R80.SECTOR]              ;init. selected sector
                   ;14606
                   ;14607 ;Fill every address of FIFO A with a fixed data byte pattern :
                   ;14608
                   ;14609 T.E.2:
U 0C2A, 2F,8295    ;14610         CLR WR[1]                                ;FIFO A address counter
U 0C2B, 17,D815    ;14611         PORT.CONTROL [SEL.FIFO.A]                ;port select FIFO A
U 0C2C, 17,C015    ;14612         PORT.CONTROL [CLEAR.FIFO.CNTR]           ;init. FIFO counter
U 0C2D, 36,9A15    ;14613         MOV LS[#55555555] TO WR[0]               ;load data pattern
                   ;14614                                                  ;only the lowest data byte is
                   ;14615                                                  ; needed for this operation
                   ;14616 FILL.FIFO.TE.1:
U 0C2E, 97,A815    ;14617         PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;start writing one byte
U 0C2F, 20,4295    ;14618         INC WR[1]                                ;inc. the address counter
                   ;14619         BIT WR[1] WITH LS[BIT10],                ;see if it gets to 400 hex yet
U 0C30, D9,54B5    ;14620          DT(LONG)&SET.ALU.CC
U 0C31, 08,C2E9    ;14621         JMP.IF[BITS.CLR] TO [FILL.FIFO.TE.1]     ;not yet, do the next byte
                   ;14622
                   ;14623 ;Now that FIFO A is full, start write data function to R80 :
```

```
                 ;14624 ;The disk address selected is head 3, sector 0 and cylinder 561 dec.
                 ;14625
                 ;14626 T.E.4:
U 0C32, B7,AC15  ;14627           MOV LS[#00046000] TO WR[0]                ;fetch cylinder 561
U 0C33, C7,4C15  ;14628           BIS LS[BIT6] TO WR[0]                     ;select head 3
U 0C34, B7,8095  ;14629           MOV LS[R80.SECTOR] TO WR[1]               ;load the current selected
U 0C35, AE,C215  ;14630           BIS WR[1] TO WR[0]                        ; sector
U 0C36, 97,A415  ;14631           PORT.WRITE PORT.ADRS[DAR] WITH WR[0]      ;load the DAR
                 ;14632
U 0C37, 37,0815  ;14633           MOV LS[WRITE.DATA] TO WR[0]               ;select write data function
U 0C38, 47,9C15  ;14634           BIS LS[SETCSR.DS0] TO WR[0]               ;select drive 0
U 0C39, 17,A015  ;14635           PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;write to the disk
                 ;14636
                 ;14637 ;************************************************************************
                 ;14638 ;* REFER TO R80 WRITE DATA IDC FUNCTION CODES AS LISTED IN TEST E ABOVE  *
                 ;14639 ;************************************************************************
                 ;14640
                 ;14641 WAIT.IDC.TE.1:
U 0C3A, DB,001A  ;14642           SKIP.IF[NO.FAST.INTERRUPT]                ;is XFER REQ set ??
U 0C3B, 08,C3D4  ;14643           JMP [IDC.DONE.TE.1]                       ;yes,
U 0C3C, 88,C3A4  ;14644           JMP [WAIT.IDC.TE.1]                       ;wait some more
                 ;14645
                 ;14646 IDC.DONE.TE.1:
U 0C3D, 90,FA15  ;14647           MISC2 [TRANSFER.GRANT]                    ;clear XFER REQ
U 0C3E, 90,9015  ;14648           MISC [SET.PORT.I.O]                       ;select from port to bus
U 0C3F, 97,8015  ;14649           PORT.READ PORT.ADRS[CSR]                  ;read the CSR
U 0C40, 3B,3215  ;14650           MOV PORT TO LS[PORTO]                     ;save the CSR word
U 0C41, 37,3215  ;14651           MOV LS[PORTO] TO WR[0]                    ;load CSR word
U 0C42, 37,C095  ;14652           MOV LS[#C00] TO WR[1]                     ;mask for OPI & DCK bits
                 ;14653           AND WR[1] TO WR[0],                       ;mask out these bits
U 0C43, AF,C235  ;14654            DT(LONG)&SET.ALU.CC                      ;CRC header error ??
U 0C44, 88,C4C9  ;14655           JMP.IF[EQL] TO [T.E.REFIL.1]              ;no, go on
U 0C45, 37,8015  ;14656           MOV LS[R80.SECTOR] TO WR[0]               ;yes, fetch sector select
U 0C46, B7,1A95  ;14657           MOV LS[#1E] TO WR[1]                      ;highest sector on a R80 track
                 ;14658           CMP WR[0] WITH WR[1],                     ;tried all sector yet ??
U 0C47, 26,40B5  ;14659            DT(LONG)&SET.ALU.CC
U 0C48, 88,C739  ;14660           JMP.IF[EQL] TO [BAD.SECTOR.TE.1]          ;report bad sector error
U 0C49, 20,4015  ;14661           INC WR[0]                                 ;try the next higher sector
U 0C4A, BF,8015  ;14662           MOV WR[0] TO LS[R80.SECTOR]
U 0C4B, 08,C2A4  ;14663           JMP [T.E.2]                               ;go write data in this sector
                 ;14664
                 ;14665 ;Refill every address of FIFO A (total = 512) with a fixed data byte pattern
                 ;14666 ; for write check function :
                 ;14667
                 ;14668 T.E.REFIL.1:
U 0C4C, 2F,8295  ;14669           CLR WR[1]                                 ;FIFO A address counter
U 0C4D, 17,D815  ;14670           PORT.CONTROL [SEL.FIFO.A]                 ;port select FIFO A
U 0C4E, 17,C015  ;14671           PORT.CONTROL [CLEAR.FIFO.CNTR]            ;init. FIFO counter
U 0C4F, 36,9A15  ;14672           MOV LS[#55555555] TO WR[0]                ;load data pattern
                 ;14673                                                     ;only the lowest data byte is
                 ;14674                                                     ; needed for this operation
                 ;14675 REFILL.FIFO.TE.1:
U 0C50, 97,A815  ;14676           PORT.WRITE PORT.ADRS[DATA.BYTE] WITH WR[0] ;start writing one byte
U 0C51, 20,4295  ;14677           INC WR[1]                                 ;inc. the address counter
                 ;14678           BIT WR[1] WITH LS[BIT10],                 ;see if it gets to 400 hex yet
```

```
U 0C52, D9,5485   ;14679          DT(LONG)&SET.ALU.CC
U 0C53, 08,C509   ;14680         JMP.IF[BITS.CLR] TO [REFILL.FIFO.TE.1]  ;not yet, do the next byte
                  ;14681
                  ;14682
                  ;14683 ;Issue the write check function here :
                  ;14684
                  ;14685 T.E.5:
U 0C54, B7,AC15   ;14686         MOV LS[#00046000] TO WR[0]              ;load DAR with disk address of
U 0C55, C7,4C15   ;14687         BIS LS[BIT6] TO WR[0]                   ;select head 3
U 0C56, 97,A415   ;14688         PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ; sec. 0, track 0 & cyl. 561
                  ;14689
U 0C57, B7,0015   ;14690         MOV LS[WRITE.CHECK] TO WR[0]            ;select write check function
U 0C58, 47,9C15   ;14691         BIS LS[SETCSR.DS0] TO WR[0]             ;select drive 0
U 0C59, 17,A015   ;14692         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;write to the disk
                  ;14693
                  ;14694 ;*** R80 Write Check IDC function codes : ***
                  ;14695
                  ;14696 ;------------------------------------------------------------------------
                  ;14697 ;=1
                  ;14698 ;
                  ;14699 ;TEST.OR.RUN:
                  ;14700 ;
                  ;14701 ;        BEN2/MAINT,             ;IS THAT YOU, VICKIE?
                  ;14702 ;        NAD/DRIVE.TYPE
                  ;14703 ;
                  ;14704 ;------------------------------------------------------------------------
                  ;14705 ;=0**
                  ;14706 ;
                  ;14707 ;DRIVE.TYPE:
                  ;14708 ;
                  ;14709 ;        BEN0/R80,               ;WHICH TYPE OF DRIVE WAS SELECTED?
                  ;14710 ;        NAD/RL02.IFORK
                  ;14711 ;
                  ;14712 ;------------------------------------------------------------------------
                  ;14713 ;=0
                  ;14714 ;
                  ;14715 ;RL02.IFORK:
                  ;14716 ;
                  ;14717 ;        UCMD/SET.FIFOA,
                  ;14718 ;        FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
                  ;14719 ;        BEN0/F0,                ;8 WAY INSTRUCTION FORK.
                  ;14720 ;        BEN1/F1,                ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                  ;14721 ;        BEN2/F2,
                  ;14722 ;        NAD/NOP.CMD
                  ;14723 ;
                  ;14724 ;------------------------------------------------------------------------
                  ;14725 ;=1
                  ;14726 ;
                  ;14727 ;R80.IFORK:
                  ;14728 ;
                  ;14729 ;        UCMD/SET.FIFOA,
                  ;14730 ;        FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
                  ;14731 ;        BEN0/F0,                ;8 WAY INSTRUCTION FORK.
                  ;14732 ;        BEN1/F1,                ;BRANCH ON F<2:0> IN THE CSR.
                  ;14733 ;        BEN2/F2,
;       NAD/#NOP.CMD
;
;-----------------------------------------------------------------------
;=001
;
;#WRT.CHK.CMD:
;
;       NAD/#WAIT.DRDY1         ;START OF THE R80 WRITE CHECK COMMAND
;
;-----------------------------------------------------------------------
;=0
;
;#WAIT.DRDY1:
;
;       BEN0/DRIVE.RDY,         ;WAIT FOR DRIVE READY
;       NAD/#WAIT.DRDY1
;
;-----------------------------------------------------------------------
;=1
;
;#CHGCLK1:
;
;       ENB.SYS.CLK,            ;USE R80 SYSTEM CLOCK
;       NAD/#WAIT.NOSEC.PLS1
;
;-----------------------------------------------------------------------
;=1*
;
;#WAIT.NOSEC.PLS1:
;
;       BEN1/SECTOR.PLS,        ;PREPARE TO LOOK FOR THE LEADING EDGE
;       NAD/#WAIT.NOSEC.PLS1    ;OF SECTOR PULSE.
;
;-----------------------------------------------------------------------
;=0*
;
;#NOSEC.PLS1:
;
;       NAD/#WAIT.SEC.PLS1      ;WE ARE NOT IN SECTOR PULSE AREA.
;
;-----------------------------------------------------------------------
;=0*
;
;#WAIT.SEC.PLS1:
;
;       BEN1/SECTOR.PLS,        ;WAIT FOR LEADING EDGE OF SEC PLS
;       NAD/#WAIT.SEC.PLS1
;
;-----------------------------------------------------------------------
;=1*
;
;#SEC.PLS.FND1:
;
;       LOAD.LOOP.CNTR.[0C5],   ;SECTOR PULSE HAS BEEN FOUND.
;       NAD/#WAIT.60.TICKS1
```

```
;14789 ;
;14790 ;-----------------------------------------------------------------------------
;14791 ;=0
;14792 ;
;14793 ;#WAIT.60.TICKS1:
;14794 ;
;14795 ;        INCR.LPCNTR/ON,              ;DELAY BEFORE ASSERTING READ GATE
;14796 ;        TAG.STROBE/ON,
;14797 ;        BENO/CNTR.OVFLW,
;14798 ;        NAD/#WAIT.60.TICKS1
;14799 ;
;14800 ;-----------------------------------------------------------------------------
;14801 ;=1
;14802 ;
;14803 ;#SETUP.LOOP.LOCK1:
;14804 ;
;14805 ;        LOOP.LOCK/ON,                ;ALLOW PHASE LOCK LOOP TO SYNC ON
;14806 ;        TAG.STROBE/ON,               ;ALL ZEROS PATTERN
;14807 ;        FUNC/CLR.ECC,
;14808 ;        LOAD.LOOP.CNTR.[0A9]
;14809 ;
;14810 ;-----------------------------------------------------------------------------
;14811 ;=0
;14812 ;
;14813 ;#WAIT.LOOP.LOCK1:
;14814 ;
;14815 ;        LOOP.LOCK/ON,                ;WAIT FOR 88 TICKS AFTER READ GATE
;14816 ;        TAG.STROBE/ON,
;14817 ;        UCON/CLR.MATCH,
;14818 ;        INCR.LPCNTR/ON,
;14819 ;        BENO/CNTR.OVFLW,
;14820 ;        NAD/#WAIT.LOOP.LOCK1
;14821 ;
;14822 ;-----------------------------------------------------------------------------
;14823 ;=1
;14824 ;
;14825 ;#SETUP.RD.HDR1:
;14826 ;
;14827 ;        LOOP.LOCK/ON,                ;SETUP INITIAL CONDITIONS
;14828 ;        TAG.STROBE/ON,               ;FOR READING HEADER.
;14829 ;        FUNC/CLR.DSR,
;14830 ;        ENB.CLR.BC/ON,
;14831 ;        LOAD.LOOP.CNTR.[0E1]
;14832 ;
;14833 ;-----------------------------------------------------------------------------
;14834 ;
;14835 ;#CHGCLK1A:
;14836 ;
;14837 ;        ENB.DSK.CLK,                 ;THE MICROSEQUENCER WILL STALL
;14838 ;        LOOP.LOCK/ON,                ;UNTIL SYNC SEEN IS SET
;14839 ;        TAG.STROBE/ON,               ;(SYNC BYTE FOUND)
;14840 ;        ENB.CLR.BC/ON,
;14841 ;        GEN.SYNC.[19]
;14842 ;
;14843 ;-----------------------------------------------------------------------------
```

```
;14844 ;=0
;14845 ;
;14846 ;#READ.HDR1:
;14847 ;
;14848 ;        LOOP.LOCK/ON,                 ;SHIFT IN 32 HEADER BITS AND
;14849 ;        TAG.STROBE/ON,                ;COMPARE WITH SERIAL DAR
;14850 ;        DATA.FIELD/ON,
;14851 ;        ENB.CLR.BC/ON,
;14852 ;        INCR.LPCNTR/ON,
;14853 ;        BENO/CNTR.OVFLW,
;14854 ;        NAD/#READ.HDR1
;14855 ;
;14856 ;------------------------------------------------------------------
;14857 ;=1
;14858 ;
;14859 ;#TEST.HDR.MATCH1:
;14860 ;
;14861 ;        UCON/TEST.MATCH,
;14862 ;        LOOP.LOCK/ON,                 ;HEADER HAS BEEN READ.
;14863 ;        TAG.STROBE/ON,                ;TURN ON TEST MATCH, OFF ENB CLR BC.
;14864 ;        DATA.FIELD/ON,                ;PREPARE TO READ THE CRC WORD.
;14865 ;        NAD/#BRANCH.HDR.MATCH1
;14866 ;
;14867 ;------------------------------------------------------------------
;14868 ;
;14869 ;#BRANCH.HDR.MATCH1:
;14870 ;
;14871 ;        LOOP.LOCK/ON,
;14872 ;        TAG.STROBE/ON,
;14873 ;        CHECK.FIELD/ON,               ;RESULTS OF HDR COMP NOW IN MISMATCH.
;14874 ;        BEN2/MISMATCH.L,
;14875 ;        NAD/#HDR.COMP1
;14876 ;
;14877 ;------------------------------------------------------------------
;14878 ;=1**
;14879 ;
;14880 ;#HDR.COMP1:
;14881 ;
;14882 ;        LOOP.LOCK/ON,                 ;HEADER COMPARE
;14883 ;        TAG.STROBE/ON,
;14884 ;        CHECK.FIELD/ON,               ;CONTINUE TO READ THE CRC WORD.
;14885 ;        LOAD.LOOP.CNTR.[OF3],
;14886 ;        BENO/CSSE,                    ;CHECK FOR SKIP SECTOR ERROR.
;14887 ;        NAD/#NOT.SSE1
;14888 ;
;14889 ;------------------------------------------------------------------
;14890 ;=0
;14891 ;
;14892 ;#NOT.SSE1:
;14893 ;
;14894 ;        LOOP.LOCK/ON,
;14895 ;        TAG.STROBE/ON,
;14896 ;        CHECK.FIELD/ON,               ;CONTINUE TO READ THE CRC WORD.
;14897 ;        NAD/#RD.CRC.WD1
;14898 ;
```

```
;14899 ;--------------------------------------------------------------------
;14900 ;=0
;14901 ;
;14902 ;@RD.CRC.WD1:
;14903 ;
;14904 ;        LOOP.LOCK/ON,               ;CONTINUE TO SHIFT IN THE 16 BIT
;14905 ;        TAG.STROBE/ON,              ;CRC WORD.
;14906 ;        CHECK.FIELD/ON,
;14907 ;        INCR.LPCNTR/ON,
;14908 ;        BEN0/CNTR.OVFLW,
;14909 ;        NAD/@RD.CRC.WD1
;14910 ;
;14911 ;--------------------------------------------------------------------
;14912 ;=1
;14913 ;
;14914 ;@CHECK.CRC1:
;14915 ;
;14916 ;        LOOP.LOCK/ON,               ;DROP CFLD. CLOCK ECC (CRC) FLOP.
;14917 ;        TAG.STROBE/ON,
;14918 ;        UCON/CHECK.ECC
;14919 ;
;14920 ;--------------------------------------------------------------------
;14921 ; This is a new state for U14IDC to allow DC007 time to
;14922 ; settle error bits.
;14923 ;
;14924 ;@CHECK.CRC2:
;14925 ;
;14926 ;        LOOP.LOCK/ON,               ;DROP CFLD. CLOCK ECC (CRC) FLOP.
;14927 ;        TAG.STROBE/ON,
;14928 ;        UCON/CHECK.ECC
;14929 ;
;14930 ;--------------------------------------------------------------------
;14931 ;
;14932 ;@TEST.FOR.ERR1:
;14933 ;
;14934 ;        LOOP.LOCK/ON,               ;WAS THERE A CRC ERROR?
;14935 ;        TAG.STROBE/ON,
;14936 ;        BEN1/ERROR.L
;14937 ;
;14938 ;--------------------------------------------------------------------
;14939 ;=1*
;14940 ;
;14941 ;@NOT.CRC.ERR1:
;14942 ;
;14943 ;        LOOP.LOCK/ON,               ;CLEAR DC007
;14944 ;        TAG.STROBE/ON,
;14945 ;        FUNC/CLR.ECC,
;14946 ;        BEN2/F2,                    ;BRANCH ON FUNCTION BITS INTO WRT CHK,
;14947 ;        BEN1/F1,                    ;RD DATA OR WRT DATA FLOWS.
;14948 ;        NAD/@BRANCH.WC.CMD1
;14949 ;
;14950 ;--------------------------------------------------------------------
;14951 ;=00*
;14952 ;
;14953 ;@BRANCH.WC.CMD1:
```

```
;14954 ;
;14955 ;        LOOP.LOCK/ON,               ;CONTINUE VIA WRITE CHECK FLOWS.
;14956 ;        TAG.STROBE/ON,
;14957 ;        BEN1/FIFO.OVFLW.L,
;14958 ;        NAD/#CHGCLK1B
;14959 ;
;14960 ;-----------------------------------------------------------------------
;14961 ;=0*
;14962 ;
;14963 ;#CHGCLK1B:
;14964 ;
;14965 ;        LOOP.LOCK/ON,               ;EVERYTHING IS GO FOR READING DATA
;14966 ;        TAG.STROBE/ON,              ;FROM THIS SECTOR. RUN OFF SYSTEM CLOCK
;14967 ;        ENB.SYS.CLK,                ;THRU HEADER POSTAMBLE AND PAST SPLICE.
;14968 ;        LOAD.LOOP.CNTR.[0F6],       ; Note: Count was changed for U14 to compensate
;14969 ;        NAD/#HDR.POST1
;14970 ;
;14971 ;-----------------------------------------------------------------------
;14972 ;=0
;14973 ;
;14974 ;#HDR.POST1:
;14975 ;
;14976 ;        TAG.STROBE/ON,              ;DROP READ GATE.
;14977 ;        INCR.LPCNTR/ON,             ;SHIFT OVER WRITE SPLICE.
;14978 ;        BEN0/CNTR.OVFLW,
;14979 ;        NAD/#HDR.POST1
;14980 ;
;14981 ;-----------------------------------------------------------------------
;14982 ;=1
;14983 ;
;14984 ;#DATA.PRE1:
;14985 ;
;14986 ;        LOOP.LOCK/ON,               ;TURN ON READ GATE AGAIN
;14987 ;        TAG.STROBE/ON,              ;NOW THAT WE'RE IN DATA PREAMBLE.
;14988 ;        LOAD.LOOP.CNTR.[0A9]
;14989 ;
;14990 ;-----------------------------------------------------------------------
;14991 ;=0
;14992 ;
;14993 ;#DATA.PRE1A:
;14994 ;
;14995 ;        LOOP.LOCK/ON,               ;ALLOW PHASE LOCK LOOP 11 BYTES
;14996 ;        TAG.STROBE/ON,              ;OF PREAMBLE TO LOCK UP.
;14997 ;        INCR.LPCNTR/ON,
;14998 ;        BEN0/CNTR.OVFLW,
;14999 ;        NAD/#DATA.PRE1A
;15000 ;
;15001 ;-----------------------------------------------------------------------
;15002 ;=1
;15003 ;
;15004 ;#SETUP.RD.DATA1:
;15005 ;
;15006 ;        LOOP.LOCK/ON,               ;SETUP INITIAL CONDITIONS
;15007 ;        TAG.STROBE/ON,              ;FOR READING DATA.
;15008 ;        UCON/CLR.MATCH,
;15009 ;        CLR.FIFO.ADR,
;15010 ;        LOAD.LOOP.CNTR.[0FB]
;15011 ;
;15012 ;-----------------------------------------------------------------
;15013 ;
;15014 ;@CHGCLK1C:
;15015 ;
;15016 ;        LOOP.LOCK/ON,             ;THE MICROSEQUENCER WILL STALL UNTIL
;15017 ;        TAG.STROBE/ON,            ;SYNC SEEN IS SET (SYNC BYTE FOUND).
;15018 ;        MODE/ECC,
;15019 ;        WRT.CHK/ON,
;15020 ;        ENB.FIFO/ON,
;15021 ;        ENB.DSK.CLK,              ;THEN WE USE READ CLOCK FROM R80.
;15022 ;        GEN.SYNC.[19]
;15023 ;
;15024 ;-----------------------------------------------------------------
;15025 ;
;15026 ;@INCR.FIFO.CNTR1:
;15027 ;
;15028 ;        LOOP.LOCK/ON,             ;ADVANCE FIFO CNTR TO REFLECT
;15029 ;        TAG.STROBE/ON,            ;HARDWARE LOAD.
;15030 ;        DATA.FIELD/ON,
;15031 ;        WRT.CHK/ON,               ;SHIFT IN 2ND BIT.
;15032 ;        MODE/ECC,
;15033 ;        INCR.FIFO.CNTR/ON
;15034 ;
;15035 ;-----------------------------------------------------------------
;15036 ;=0
;15037 ;
;15038 ;@READ.DATA1:
;15039 ;
;15040 ;        LOOP.LOCK/ON,             ;SHIFT IN 5 DATA BITS.
;15041 ;        TAG.STROBE/ON,
;15042 ;        DATA.FIELD/ON,
;15043 ;        WRT.CHK/ON,
;15044 ;        MODE/ECC,
;15045 ;        INCR.LPCNTR/ON,
;15046 ;        BENO/CNTR.OVFLW,
;15047 ;        NAD/@READ.DATA1
;15048 ;
;15049 ;-----------------------------------------------------------------
;15050 ;=1
;15051 ;
;15052 ;@LOAD.BYTE1:
;15053 ;
;15054 ;        LOOP.LOCK/ON,             ;8TH DATA BIT
;15055 ;        TAG.STROBE/ON,
;15056 ;        DATA.FIELD/ON,
;15057 ;        WRT.CHK/ON,
;15058 ;        ENB.FIFO/ON,
;15059 ;        LOAD.DSR/ON,              ;LOAD NEXT BYTE FROM FIFO INTO DSR.
;15060 ;        MODE/ECC,
;15061 ;        LOAD.LOOP.CNTR.[0F9],
;15062 ;        INCR.FIFO.CNTR/ON,
;15063 ;        NAD/@READ.DATA1A
```

```
;
;-------------------------------------------------------------------------------
;=10
;
;#READ.DATA1A:
;
;       LOOP.LOCK/ON,               ;SHIFT IN 7 DATA BITS.
;       TAG.STROBE/ON,
;       DATA.FIELD/ON,
;       WRT.CHK/ON,
;       MODE/ECC,
;       INCR.LPCNTR/ON,
;       BEN0/CNTR.OVFLW,
;       NAD/#READ.DATA1A
;
;-------------------------------------------------------------------------------
;=11
;
;#LOAD.BYTE1A:
;
;       LOOP.LOCK/ON,               ;8TH DATA BIT
;       TAG.STROBE/ON,
;       DATA.FIELD/ON,
;       WRT.CHK/ON,
;       ENB.FIFO/ON,
;       LOAD.DSR/ON,                ;LOAD NEXT BYTE FROM FIFO INTO DSR.
;       MODE/ECC,
;       LOAD.LOOP.CNTR.[0F9],
;       INCR.FIFO.CNTR/ON,
;       BEN1/FIFO.OVFLW.L,
;       NAD/#READ.DATA1A
;
;-------------------------------------------------------------------------------
;=00
;
;#RD.DATA.ECC1:
;
;       LOOP.LOCK/ON,               ;PREPARE TO READ THE ECC WORD
;       TAG.STROBE/ON,
;       DATA.FIELD/ON,
;       MODE/ECC,
;       UCON/TEST.MATCH,            ;STROBE RESULTS OF DATA XOR.
;       LOAD.LOOP.CNTR.[0E0],
;       NAD/#RD.DATA.ECC1A
;
;-------------------------------------------------------------------------------
;=0
;
;#RD.DATA.ECC1A:
;
;       LOOP.LOCK/ON,               ;SHIFT IN 32 BIT FIRE CODE
;       TAG.STROBE/ON,
;       CHECK.FIELD/ON,
;       MODE/ECC,
;       INCR.LPCNTR/ON,
;         BENO/CNTR.OVFLW,
;         NAD/#RD.DATA.ECC1A
;
;------------------------------------------------------------------------
;=1
;
;#CHECK.ECC1:
;
;         LOOP.LOCK/ON,             ;DROP CFLD. CLOCK ECC FLOP.
;         TAG.STROBE/ON,
;         MODE/ECC,
;         UCON/CHECK.ECC
;
;------------------------------------------------------------------------
; Extra state inserted for U14IDC to allow DC007 time to settle
;
;#CHECK.ECC2:
;
;         LOOP.LOCK/ON,             ;DROP CFLD. CLOCK ECC FLOP.
;         TAG.STROBE/ON,
;         MODE/ECC,
;         UCON/CHECK.ECC
;
;------------------------------------------------------------------------
;
;#TEST.FOR.ERR1A:
;
;         LOOP.LOCK/ON,
;         TAG.STROBE/ON,            ;SWITCH OVER TO CPU CLOCK
;         ENB.CPU.CLK,              ;FOR FAST INTERRUPT OR TERMINATE.
;         BEN1/ERROR.L              ;WAS THERE AN ECC OR WRT CHK ERROR?
;
;------------------------------------------------------------------------
;=1*
;
;#NOT.ECC.ERR1:
;
;         CLR.FIFO.ADR,             ;CLEAR THE FIFO ADDRESS COUNTER.
;         UCMD/SET.XFER.RQST,
;         NAD/#WAIT.XFER.GRANT1
;
;------------------------------------------------------------------------
;=1*
;
;#WAIT.XFER.GRANT1:
;
;         BEN1/XFER.REQ,            ;CPU WILL TURN ON CRDY IF IT'S THE LAST
;         NAD/#WAIT.XFER.GRANT1     ;SECTOR, THEN SEND GRANT.
;
;------------------------------------------------------------------------
;=0*
;
;#TEST.XFER.DONE1:
;
;         UCMD/CHG.FIFO,            ;SWITCH FIFO'S, IN CASE THERE IS MORE
```

```
;15174 ;        BENO/CRDY.L,                ;DATA TO BE READ.
;15175 ;        NAD/#XFER.DONE1
;15176 ;
;15177 ;-------------------------------------------------------------------
;15178 ;=0
;15179 ;
;15180 ;#XFER.DONE1:
;15181 ;
;15182 ;        UCMD/SET.CRDY,
;15183 ;        INCR.DAR/ON,
;15184 ;        FUNC/SET.INT,               ;CPU SIGNALS END OF TRANSFER.
;15185 ;        NAD/SET.INT.AND.IDLE
;15186 ;
;15187 ;-------------------------------------------------------------------
;15188 ;=111111110
;15189 ;
;15190 ;SET.INT.AND.IDLE:
;15191 ;
;15192 ;        FUNC/SET.INT,               ; The purpose of this state is to give CRDY one
;15193 ;        NAD/IDLE                    ; state to reset before asserting interrupt.
;15194 ;                                    ; CRDY is now used in the INT equation.
;15195 ;
;15196 ;-------------------------------------------------------------------
;15197 ;=1
;15198 ;
;15199 ;RTN.TO.IDLE:
;15200 ;IDLE:
;15201 ;
;15202 ;        BENO/CRDY.L,                ;LOOK FOR GO (CRDY L)
;15203 ;        NAD/SETUP.DRV.CHK
;15204 ;
;15205 ;-------------------------------------------------------------------
;15206 ;=0
;15207 ;
;15208 ;SETUP.DRV.CHK:
;15209 ;
;15210 ;        LOAD.LOOP.CNTR.[OFB],       ;COUNTER WILL BE USED
;15211 ;        NAD/SEL.NXT.DRV             ;TO SEQUENCE THRU DRIVES.
;15212 ;
;15213 ;-------------------------------------------------------------------
;15214 ;
;15215 ;SEL.NXT.DRV:
;15216 ;
;15217 ;        SEL.DRV/UPROG,
;15218 ;        BENO/CNTR.OVFLW,
;15219 ;        NAD/CHK.DRV.STATUS
;15220 ;
;15221 ;-------------------------------------------------------------------
;15222 ;=0
;15223 ;
;15224 ;CHK.DRV.STATUS:
;15225 ;        ; Give drive status time to settle down.
;15226 ;
;15227 ;        SEL.DRV/UPROG,              ;SELECT DRIVE
;15228 ;        NAD/DRV.STATUS.DISPATCH
```

```
                    ;15229 ;
                    ;15230 ;-----------------------------------------------------------------
                    ;15231 ; BEN2/DRIVE.ERR was removed from this state to prevent VMS hang
                    ;15232 ;
                    ;15233 ;DRV.STATUS.DISPATCH:
                    ;15234 ;        SEL.DRV/UPROG,          ;SELECT DRIVE
                    ;15235 ;        BEN1/DRIVE.ONLINE,      ;AND BRANCH ON STATUS.
                    ;15236 ;        BEN0/DRIVE.RDY,
                    ;15237 ;        NAD/-ERR.-OL.-RDY
                    ;15238 ;
                    ;15239 ;
                    ;15240 ;-----------------------------------------------------------------
                    ;15241 ;=011
                    ;15242 ;
                    ;15243 ;-ERR.OL.RDY:
                    ;15244 ;
                    ;15245 ;        SEL.DRV/UPROG,          ;NOP
                    ;15246 ;        INCR.LPCNTR/ON,         ;ADD 1 TO DRIVE SELECT.
                    ;15247 ;        NAD/SEL.NXT.DRV
                    ;15248 ;
                    ;15249 ;-----------------------------------------------------------------
                    ;15250 ;
                    ;15251 ; *** KEEP LOOPING FROM  "SEL.NXT.DRV:"   THROUGH   "-ERR.OL.RDY:"   ***
                    ;15252 ;
                    ;15253 ;               UNTIL "CNTR.OVFLW" IS SET, THEN "NAD/IDLE"
                    ;15254 ;
                    ;15255 ;-----------------------------------------------------------------
                    ;15256 ;
                    ;15257 ;=1
                    ;15258 ;
                    ;15259 ;RTN.TO.IDLE:
                    ;15260 ;IDLE:
                    ;15261 ;
                    ;15262 ;        BEN0/CRDY.L,            ;LOOK FOR GO (CRDY L)
                    ;15263 ;        NAD/SETUP.DRV.CHK
                    ;15264 ;
                    ;15265 ;-----------------------------------------------------------------
                    ;15266
                    ;15267
                    ;15268 ;Set up parameters for time-out error report :
                    ;15269
U 0C5A, 36,4095     ;15270          MOV LS[#1] TO WR[1]
U 0C5B, 3E,8295     ;15271          MOV WR[1] TO LS[ERROR.NUMBER]           ;set up error number = 1 in LS
U 0C5C, E5,8A15     ;15272          CLR LS[ERROR.MASK]                      ;no error mask used here
U 0C5D, 36,4A95     ;15273          MOV LS[IDC] TO WR[1]                    ;store module code in LS to
U 0C5E, BE,8C95     ;15274          MOV WR[1] TO LS[MODULE.NUM]             ; modules
U 0C5F, 36,6E95     ;15275          MOV LS[NA.EXP.REC] TO WR[1]             ;put "N/A" for printout data
U 0C60, BE,8095     ;15276          MOV WR[1] TO LS[CONTROL.STATUS]
                    ;15277
U 0C61, 2F,8015     ;15278          CLR WR[0]                               ;Initialize time-out counter
U 0C62, BF,8C15     ;15279          MOV WR[0] TO LS[TIMEOUT.CNTR]
                    ;15280
                    ;15281 WAIT.IDC.TE.2:
U 0C63, 37,8C15     ;15282          MOV LS[TIMEOUT.CNTR] TO WR[0]           ;load the current time count
U 0C64, 37,9095     ;15283          MOV LS[WAIT.TIME1] TO WR[1]             ;time limit on write check
```

```
                 ;15284          CMP WR[0] WITH WR[1],                    ;timeout yet ???
U 0C65, 26,4085  ;15285           DT(LONG)&SET.ALU.CC
U 0C66, 08,C6C9  ;15286          JMP.IF[EQL] TO [TE.ERR.1]                ;yes, timeout on write check
U 0C67, 20,4015  ;15287          INC WR[0]                                ;inc. time count
U 0C68, BF,8C15  ;15288          MOV WR[0] TO LS[TIMEOUT.CNTR]
U 0C69, DB,001A  ;15289          SKIP.IF[NO.FAST.INTERRUPT]               ;check if XFER REQ is set
U 0C6A, 88,C714  ;15290          JMP [IDC.DONE.TE.2]                      ;yes, XFER REQ is set now
U 0C6B, 88,C634  ;15291          JMP [WAIT.IDC.TE.2]                      ;no, wait some more
                 ;15292
                 ;15293 ;Force an error in the CHECK.RESULT routine for this timout :
                 ;15294
                 ;15295 TE.ERR.1:
U 0C6C, 2F,8015  ;15296          CLR WR[0]                                ;set unequal values
U 0C6D, 36,4095  ;15297          MOV LS[01] TO WR[1]                      ; between WR0 and WR1
U 0C6E, 08,693C  ;15298          JSR [CHECK.RESULT]                       ;force an error
U 0C6F, 08,C544  ;15299          JMP [T.E.5]                              ;err and if LOE is set
U 0C70, 08,C3D4  ;15300          JMP [IDC.DONE.TE.1]
                 ;15301
                 ;15302 IDC.DONE.TE.2:
U 0C71, 90,FA15  ;15303          MISC2 [TRANSFER.GRANT]                   ;clear XFER REQ
U 0C72, 88,C7E4  ;15304          JMP [END.TE]                             ;test ends
                 ;15305
                 ;15306 ;Could not find a good sector on the entire track, this section of codes will
                 ;15307 ; force an error and report it :
                 ;15308
                 ;15309 BAD.SECTOR.TE.1:
U 0C73, B6,4295  ;15310          MOV LS[02] TO WR[1]
U 0C74, 3E,8295  ;15311          MOV WR[1] TO LS[ERROR.NUMBER]            ;set up error number = 2 in LS
U 0C75, E5,8A15  ;15312          CLR LS[ERROR.MASK]                       ;no error mask used here
U 0C76, B6,4E95  ;15313          MOV LS[R80] TO WR[1]                     ;store module code in LS to
U 0C77, BE,8C95  ;15314          MOV WR[1] TO LS[MODULE.NUM]              ; modules
U 0C78, 36,6E95  ;15315          MOV LS[NA.EXP.REC] TO WR[1]              ;put "N/A" for printout data
U 0C79, BE,8095  ;15316          MOV WR[1] TO LS[CONTROL.STATUS]
                 ;15317
                 ;15318 ;Force an error to indicate that there are no good sector found on this track :
                 ;15319
U 0C7A, 2F,8015  ;15320          CLR WR[0]                                ;set received data and
U 0C7B, 36,4095  ;15321          MOV LS[01] TO WR[1]                      ; expected data unequal
U 0C7C, 08,693C  ;15322          JSR [CHECK.RESULT]                       ;forcing the error
U 0C7D, 08,C2A4  ;15323          JMP [T.E.2]                              ;err and if LOE is set
                 ;15324
                 ;15325 END.TE:                                          ;end of test
U 0C7E, 08,E064  ;15326          JMP [END.SECTION]                        ;done with standard tests
                 ;15327
                 ;15328
```

```
;15329 .PAGE  "Test F  TEST FOR THE RIGHT CRD VOLUME LABEL ON RL02(M8388 IDC MODULE)"
;15330 ;
;15331 ;*** The following tests " F & 10 " are optional for CRD ***
;15332 ;
;15333 ; Test Description:
;15334 ;
;15335 ;        This test will first check for a RL02 drive 1 being ready by
;15336 ;        selecting DS bits in the CSR to the drive and then reading the
;15337 ;        CSR back checking for DRDY bit set and R80 bit reset. If this
;15338 ;        is not the case the test will wait about 11 seconds for the
;15339 ;        drive to settle. Should this still fail to get the correct CSR
;15340 ;        word, then the message "PROPER SETUP REQUIRED - CHECK FOR RL02
;15341 ;        DIAGNOSTIC PACK 'CRDPACK01' INSERTED AND SPINNING, IN DRIVE
;15342 ;        DQA1. IF SETUP IS CORRECT, SUSPECT RL02 DRIVE PROBLEM : - THEN
;15343 ;        TYPE <CR> TO CONTINUE" will be issued. It will next exercise the
;15344 ;        get status function to determine if the drive status is good so
;15345 ;        that the remaining test can proceed. The status bits are
;15346 ;        categorized as follows: Hardware errors (ones that are originated
;15347 ;        from failure of the hardware) and Setup errors (ones that are most
;15348 ;        likely originated from incorrect drive setup). The following table
;15349 ;        summarizes different status bit error type.
;15350 ;
;15351 ;                  Note, the state bits <2:0> are checked first. If these bits
;15352 ;        do not indicate the expected LOCK ON state, the program waits for
;15353 ;        a yet-to-be-determined time and then rechecks state. After this
;15354 ;        second check if the state is still not LOCK ON then the indicated
;15355 ;        error type is issued. The other status bits are checked only if the
;15356 ;        state has become locked on.
;15357 ;
;15358 ;                                              set
;15359 ;                  RL02 drive 1   HDW   -up   expected
;15360 ; Functions        status bits    err   err   states
;15361 ; ----------       ------------   ---   ---   --------
;15362 ;
;15363 ; load cart.       <0:2>  000            x    101 lock on
;15364 ; spin up          <0:2>  001      x          101 lock on
;15365 ; brush cycle      <0:2>  010      x          101 lock on
;15366 ; load heads       <0:2>  011      x          101 lock on
;15367 ; seek             <0:2>  100      x          101 lock on
;15368 ; lock on          <0:2>  101                 101 lock on
;15369 ; unload heads     <0:2>  110      x          101 lock on
;15370 ; spin down        <0:2>  111      x          101 lock on
;15371 ;
;15372 ; brush home         <3>  ---      x            1 brush off disk
;15373 ; heads out          <4>  ---      x            1 heads on disk
;15374 ; cover open         <5>  ---      x            0 cover closed
;15375 ; head select        <6>  ---      x            0 upper head select
;15376 ; not used           <7>  ---                   1 (will be masked out)
;15377 ; dr. sel. err.      <8>  ---      x            0 only one drive responded
;15378 ; volume check       <9>  ---      x            0 cartridge not changed
;15379 ; write g. err.     <10>   ---      x            0 not doing a write
;15380 ; spin error        <11>   ---      x            0 spindle speed correct
;15381 ; seek timeout      <12>   ---      x            0 not doing a seek
;15382 ; write lock        <13>   ---                   X don't care
;15383 ; hd. current       <14>   ---      x            0 not doing a write
```

```
;15384 ; wrt. dat. err. <15>  ---      x          0 not doing a write
;15385 ;
;15386 ;
;15387 ;       With Hardware error Auto test will report device failure
;15388 ;       and issue the message "AUTO TEST ABORTED - IDC BAD"; with setup
;15389 ;       error Auto test will issue the warning message to the console
;15390 ;       and wait for the appropiate response before proceeding.
;15391 ;
;15392 ;       Next this test will find out where the heads are located by the
;15393 ;       "RL READ HEADER" function by first : (write to the CSR with
;15394 ;       drive 1 select <bit 8> and the read header function <bit 3>. Then
;15395 ;       the header information will be read off FIFO A). Next, a seek
;15396 ;       function to a predetermined cylinder will be performed  :  (the
;15397 ;       cylinder difference which will be loaded to the DAR with the marker,
;15398 ;       direction and cylinder difference bits; then CSR will be written
;15399 ;       with drive select 1 <bit 8> and the seek function <bits 1 and 2>).
;15400 ;       Then to verify that read header works and the heads are really at
;15401 ;       cylinder zero, a read header command will be issued to the drive to
;15402 ;       obtain the current cylinder address which will be compared against
;15403 ;       the expected address (cylinder = 0 in this case). If the cylinder
;15404 ;       address is incorrect a hardware error will be reported.
;15405 ;
;15406 ;       Then this test will read the label by the READ DATA function to the
;15407 ;       home_block resident on cylinder zero, sector 4. It will then write to
;15408 ;       the DAR with the proper disk address format <bits 0 to 15>, sector 4,
;15409 ;       head 0 and cylinder 0. Then the CSR will be written with drive select
;15410 ;       1 <bit 8> and the read data function <bits 2 and 3>. The label name
;15411 ;       located from the 216th to the 227th bytes of the 256 home_block bytes
;15412 ;       will be read off from FIFO A and compared with the expected label
;15413 ;       "CRDPACK01" in the local stores. If the label does not compare, the
;15414 ;       setup warning message will be issued to the console and wait for an
;15415 ;       appropriate response from the operator.
;15416 ;
;15417 ;       Note that if the homeblock becomes a bad block, the test will
;15418 ;       search for a valid backup homeblock in the first 2 cylinders
;15419 ;       to obtain the label. If no valid homeblock is found, the setup
;15420 ;       warning message will be issued to the console.
;15421 ;
;15422 ; Assumption:
;15423 ;
;15424 ;       Both CPU micros and IDC part 1 has been run successfully
;15425 ;
;15426 ; Test Steps:
;15427 ;
;15428 ;     1. Load the hex equivalent for label "CRDPACK      "
;15429 ;        and then, initialize the wait time counter
;15430 ;     2. Write CSR by setting its drive 1 selection bit <8>
;15431 ;     3. Set up CSR error report parameters
;15432 ;     4. Read the CSR word back
;15433 ;     5. Mask this CSR word except its R80 & DRDY bits <26 & 0>
;15434 ;     6. Check the R80 bit for reset & DRDY set; if not
;15435 ;     7.    Then check wait time time-out, if so
;15436 ;     8.         Then issue the WARNING MESSAGE;
;15437 ;                     wait for console response before going
;15438 ;                     back to beginning of test
```

```
;     9.         Else inc. wait time counter & loop back to step 2
;    10.    Else proceed
;    11. Clear the wait time counter (i.e. LS[WAIT.CNTR] )
;    12. Write DAR by setting its marker,status and reset bits <3, 1 & 0>
;    13. Write CSR with drive 1 select & get status bits <8 & 2>
;    14. Wait for CRDY set
;    15. Read the RL02 status from FIFO A
;    16. Mask RL02 status with LS[#00000007] for the state bits
;    17. Check the status whether its state is lock on <=101>; if not
;    18. Then check whether state is zero; if not
;    19.      Then does LS[WAIT.CNTR] exceeds max. wait time? If not
;    20.           Then inc. LS[WAIT.CNTR] & loop back to step 12
;    21.           Else report the hardware ERROR 1
;    22.      Else report the WARNING MESSAGE and wait for console
;                 response before going back to the beginning of test
;    23. Else proceed
;    24. End check
;    25. Read the drive status again from FIFO A
;    26. Load LS[#FFFF20C7] into LS[ERROR.MASK]
;    27. Compare the remaining status bits <15:3> with the expected
;          LS[#00000018] by JSR [CHECK.RESULT]
;    28. If they are not equal; Then report hardware ERROR 2
;    29. Else Write CSR with drive 1 select and read header function bits
;          <8 & 3>
;    30. Wait for CRDY to set
;    31. Port select FIFO A and clear its address counter
;    32. Read header from FIFO A
;    33. Mask the header with LS[#FFFFFF80] to get the cylinder address
;    34. Write DAR with dir bit and marker bit set <2 & 0>and the cylinder
;          address difference bits <15:7> (current cyl. adr. minus zero)
;    35. Write CSR with drive 1 select and seek function bits
;          <8, 2 & 1>
;    36. Wait for CRDY to set
;    37. Write CSR with drive 1 select and read header function bits
;    38. Wait for CRDY set
;    39. Port select FIFO A and clear its address counter
;    40. Read header from FIFO A
;    41. Load LS[ERROR.MASK] with LS[#0000007F] for the cylinder difference
;    42. Compare the cylinder address with zero by JSR [CHECK.RESULT]
;    43. If they don't compare; report ERROR 3
;    44. Else initialize DAR adr. bits to sector 3, track 0 & cyl. 0
;    45. Write DAR with the current DAR adr. bit format
;    46. Write CSR with drive 1 select and read data function bits
;          <8, 3 & 2>
;    47. Wait for XFER REQ
;    48. Send TRANFER GRANT to clear the XFER REQ
;    49. Port select FIFO A and clear its address counter
;    50. Increment FIFO A address counter by 1
;    51. Read the word from FIFO A into WR[0]
;    52. Check WR[0] for zero content (0=invalid homeblock), if so
;    53.     Then check if sector number has reached 39(dec) yet,if so
;    54.          Then check if cyl. is less than 2, if so
;    55.               Then increment cyl. by 1;
;                             reset sector to 0 in DAR adr. bits
;                             and go back to step 45
```

```
;15494 ;   56.             Else issue the WARNING MESSAGE and wait for
;15495 ;                        console response before going back to
;15496 ;                        beginning of test
;15497 ;   57.         Else inc. sector by 2 in DAR adr. bits and go back
;15498 ;                        to step 45
;15499 ;   58.     Else proceed
;15500 ;   59. Increment sector by 1 in the DAR adr. bits
;15501 ;   60. Write DAR with the current adr. bit format
;15502 ;   61. Write CSR with drive 1 select read data function bits
;15503 ;       <8, 3 & 2>
;15504 ;   62. Wait for XFER REQ
;15505 ;   63. Send TRANSFER GRANT to clear the XFER REQ
;15506 ;   64. Port select FIFO A and clear its address counter
;15507 ;   65. Step up FIFO A address counter until it gets to the beginning
;15508 ;       of the 12-byte label (from D8 to E3 hex) and read it into 3
;15509 ;       LS locations by using JSR [LONG.WORD.LABEL] for comparison
;15510 ;       later
;15511 ;   66. Retrieve the label from the LS locations and compare that
;15512 ;       with the hex equivalent (4352445041434B2020202020) of the
;15513 ;       label "CRDPACK     "
;15514 ;   67. If data not compare
;15515 ;   68.        Then report the WARNING MESSAGE and wait for console
;15516 ;                   response before going back to beginning of test
;15517 ;   69.        Else Test ends
;15518 ;
;15519 ;
;15520 ;
;15521 ; WARNING MESSAGE:
;15522 ;
;15523 ;        "PROPER SETUP REQUIRED - CHECK FOR RL02 DIAGNOSTIC PACK
;15524 ;         'CRDPACK01' INSERTED AND SPINNING, IN DRIVE DQA1. IF
;15525 ;         SETUP IS CORRECT, SUSPECT RL02 DRIVE PROBLEM : - THEN
;15526 ;         TYPE <CR> TO CONTINUE"
;15527 ;
;15528 ;
;15529 ; ERRORS:
;15530 ;
;15531 ; ERROR 1 - failed to find a RL02 up and spinning in drive 1
;15532 ;
;15533 ;          Printout:
;15534 ;               EXPECTED         RECEIVED          MODULE
;15535 ;               CSR<26 & 0>      CSR<26 & 0>       RL02
;15536 ;
;15537 ; ERROR 2 - RL02 drive 1 status state bits failed
;15538 ;
;15539 ;          Printout:
;15540 ;               EXPECTED            RECEIVED            MODULE
;15541 ;               RL02 status         RL02 status         IDC/RL02
;15542 ;               state bits          state bits
;15543 ;
;15544 ; ERROR 3 - RL02 drive 1 status bits failed
;15545 ;
;15546 ;          Printout:
;15547 ;               EXPECTED              RECEIVED              MODULE
;15548 ;               RL02 remaining        RL02 remaining        IDC/RL02
```

```
                   ;15549 ;               status bits         status bits
                   ;15550 ;
                   ;15551 ; ERROR 4 - RL02 drive 1 R/W heads did not seek to the outermost cylinder
                   ;15552 ;
                   ;15553 ;         Printout:
                   ;15554 ;              EXPECTED         RECEIVED        MODULE
                   ;15555 ;              cyl. adr.        cyl. adr.       IDC/RL02
                   ;15556 ;
                   ;15557 ; ERROR 5 - failed to find a valid homeblock on the first 2 cylinders of
                   ;15558 ;            the disk
                   ;15559 ;
                   ;15560 ;         Printout:
                   ;15561 ;              EXPECTED         RECEIVED        MODULE
                   ;15562 ;              N/A              N/A             RL02
                   ;15563 ;
                   ;15564 ; ERROR 6 - a correct CRD pack label was not found in RL02 drive 1
                   ;15565 ;
                   ;15566 ;         Printout:
                   ;15567 ;              EXPECTED            RECEIVED
                   ;15568 ;              RL02 pack           RL02 pack
                   ;15569 ;              label (hex)         label (hex)
                   ;15570 ; Debug:
                   ;15571 ;
                   ;15572 ;   Errors will occur if any of the status bits is stuck fault
                   ;15573 ;   or if disk drive 1 is not setup for operational status to
                   ;15574 ;   receive commands or its hardware is at fault
                   ;15575 ;
                   ;15576 ;--
                   ;15577
                   ;15578 T.F:
U 0C7F, B6,5E15    ;15579         MOV LS[BEGIN.TEST] TO WR[0]              ;set bit 15 in WR[0] for
                   ;15580                                                  ; control and status word
U 0C80, 3E,8015    ;15581         MOV WR[0] TO LS[CONTROL.STATUS]          ;set bit 15 in control and
                   ;15582                                                  ; status word - bit 15
                   ;15583                                                  ; indicates start of test to
                   ;15584                                                  ; console
U 0C81, 10,E015    ;15585         MISC [SET.CP.ATTN]                       ;issue CPU ATTN to console
                   ;15586
                   ;15587 WAIT.TF.0:
U 0C82, 88,C824    ;15588         JMP [WAIT.TF.0]                          ;loop here waiting for
                   ;15589                                                  ; console to respond
                   ;15590
                   ;15591 ;Store the LS[LABEL.SAV'S] here so that error report will print out ASCII instead
                   ;15592 ; of hex values :
                   ;15593
U 0C83, B7,2215    ;15594         MOV LS[#43524450] TO WR[0]               ;hex value for "CRDP"
U 0C84, 3F,1415    ;15595         MOV WR[0] TO LS[LABEL.SAV1]              ; to be printed upon error
U 0C85, B7,2415    ;15596         MOV LS[#41434B20] TO WR[0]               ;hex value for "ACK "
U 0C86, BF,1615    ;15597         MOV WR[0] TO LS[LABEL.SAV2]              ; to be printed upon error
U 0C87, 37,2C15    ;15598         MOV LS[#20202020] TO WR[0]               ;hex value for "    "
U 0C88, 3F,1815    ;15599         MOV WR[0] TO LS[LABEL.SAV3]              ; to be printed upon error
                   ;15600
U 0C89, 65,A015    ;15601         CLR LS[PREVIOUS.ERROR]                   ;clear previous error number
                   ;15602                                                  ; in LS
U 0C8A, 17,CC15    ;15603         PORT.CONTROL [CLEAR.IDC]                 ;init. IDC
```

```
U 0C8B, 17,CC15   ;15604          PORT.CONTROL [CLEAR.IDC]
U 0C8C, 17,CC15   ;15605          PORT.CONTROL [CLEAR.IDC]
                  ;15606
                  ;15607 TEST.INIT.1:
U 0C8D, 2F,8015   ;15608          CLR WR[0]
U 0C8E, 3F,8E15   ;15609          MOV WR[0] TO LS[WAIT.CNTR]              ;init. wait loop counter
                  ;15610
                  ;15611 TEST.TF.1:
U 0C8F, 36,5015   ;15612          MOV LS[SETCSR.DS1] TO WR[0]             ;drive 1 select bit
U 0C90, 17,A015   ;15613          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;write CSR to select dr. 1
                  ;15614
U 0C91, B7,3015   ;15615          MOV LS[#1A] TO WR[0]                    ;set up a counter to nop until
                  ;15616 NOP.TF.1:
U 0C92, 21,0015   ;15617          DEC WR[0]                               ; CSR has finished writing
                  ;15618          CMP WR[0] WITH LS[ZERO],
U 0C93, 4F,9C35   ;15619           DT(LONG)&SET.ALU.CC
U 0C94, 08,C921   ;15620          JMP.IF[NEQ] TO [NOP.TF.1]
                  ;15621
U 0C95, 36,4095   ;15622          MOV LS[#1] TO WR[1]                     ;set WR1 = 1
U 0C96, 3E,8295   ;15623          MOV WR[1] TO LS[ERROR.NUMBER]           ;set up error number = 1 in LS
U 0C97, 36,4A95   ;15624          MOV LS[IDC] TO WR[1]                    ;store module code in LS
U 0C98, 47,4C95   ;15625          BIS LS[RL02] TO WR[1]                   ; to indicate IDC and RL02
U 0C99, BE,8C95   ;15626          MOV WR[1] TO LS[MODULE.NUM]             ; modules
U 0C9A, B7,9E95   ;15627          MOV LS[#FBFFFFFE] TO WR[1]              ;error mask for CSR word
U 0C9B, BE,8A95   ;15628          MOV WR[1] TO LS[ERROR.MASK]
U 0C9C, B6,4915   ;15629          MOV LS[BIT4] TO WR[2]                   ;set bit4 for CRD micmon
U 0C9D, BE,8115   ;15630          MOV WR[2] TO LS[CONTROL.STATUS]         ; for printing of error msg.
                  ;15631
U 0C9E, 90,9015   ;15632          MISC [SET.PORT.I.0]                     ;select the port to bus
U 0C9F, 97,8015   ;15633          PORT.READ PORT.ADRS[CSR]                ;read CSR back
U 0CA0, 3B,3215   ;15634          MOV PORT TO LS[PORT0]                   ; from the port and
U 0CA1, 37,3215   ;15635          MOV LS[PORT0] TO WR[0]                  ; load it in WR0
U 0CA2, 37,AA95   ;15636          MOV LS[#04000001] TO WR[1]              ;mask for bits <26 & 0>
U 0CA3, 2F,C215   ;15637          AND WR[1] TO WR[0]                      ;mask out WR[0]
U 0CA4, 36,4095   ;15638          MOV LS[BIT0] TO WR[1]                   ;load DRDY bit in WR[1]
                  ;15639          CMP WR[0] WITH WR[1],                   ;RL02 in drive 1 ?
U 0CA5, 26,4085   ;15640           DT(LONG)&SET.ALU.CC
U 0CA6, 88,CB39   ;15641          JMP.IF[EQL] TO [GET.STATUS.1]           ;yes, do the next test
                  ;15642
U 0CA7, 37,8E95   ;15643          MOV LS[WAIT.CNTR] TO WR[1]              ;inspect the wait loop counter
U 0CA8, 37,9115   ;15644          MOV LS[WAIT.TIME1] TO WR[2]             ;get the wait time
                  ;15645          CMP WR[1] WITH WR[2],                   ;exceed wait time limit ?
U 0CA9, A6,4335   ;15646           DT(LONG)&SET.ALU.CC
U 0CAA, 20,4295   ;15647          INC WR[1]                               ;increment the
U 0CAB, BF,8E95   ;15648          MOV WR[1] TO LS[WAIT.CNTR]              ; wait counter by 1
U 0CAC, 58,0009   ;15649          SKIP.IF[EQL]                            ;timeout, go print the error
U 0CAD, 08,D844   ;15650          JMP [WRONG.LABEL1]                      ;give RL02 more time to settle
                  ;15651
U 0CAE, 08,D844   ;15652          jmp [wrong.label1]                      ;flag to CRD micmon
                  ;15653
                  ;15654 WAIT.TF.1:
U 0CAF, 88,CAF4   ;15655          JMP [WAIT.TF.1]                         ;wait for console to respond
U 0CB0, 36,4095   ;15656          MOV LS[BIT0] TO WR[1]                   ;load DRDY bit as exp. data
U 0CB1, 08,693C   ;15657          JSR [CHECK.RESULT]                      ;enter here to get a standard
                  ;15658                                                  ; error report for reg. micmon
```

```
U OCB2, 88,C8D4   ;15659          JMP [TEST.INIT.1]                        ;loop on err if err bit is set
                  ;15660
                  ;15661 GET.STATUS.1:
U OCB3, 2F,8015   ;15662          CLR WR[0]
U OCB4, 3F,8E15   ;15663          MOV WR[0] TO LS[WAIT.CNTR]              ;init. wait loop counter
                  ;15664
                  ;15665 TEST.TF.2:
U OCB5, 37,C415   ;15666          MOV LS[#B] TO WR[0]                     ;DAR format for get status
U OCB6, 97,A415   ;15667          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ;write DAR with the command
                  ;15668
U OCB7, B7,0295   ;15669          MOV LS[GET.STATUS] TO WR[1]             ;CSR format for get status
U OCB8, 36,5015   ;15670          MOV LS[SETCSR.DS1] TO WR[0]             ;select drive 1
U OCB9, AE,C095   ;15671          BIS WR[0] TO WR[1]
U OCBA, 97,A095   ;15672          PORT.WRITE PORT.ADRS[CSR] WITH WR[1]    ;write to the CSR
                  ;15673                                                  ;MAINT=0,CRDY=0,F<2:0>=010
                  ;15674
                  ;15675 ;*** RL02 GET STATUS IDC FUNCTION CODES : ***
                  ;15676
                  ;15677 ;------------------------------------------------------------------
                  ;15678 ;=1
                  ;15679 ;
                  ;15680 ;TEST.OR.RUN:
                  ;15681 ;
                  ;15682 ;        BEN2/MAINT,             ;IS THAT YOU, VICKIE?
                  ;15683 ;        NAD/DRIVE.TYPE
                  ;15684 ;
                  ;15685 ;------------------------------------------------------------------
                  ;15686 ;=0**
                  ;15687 ;
                  ;15688 ;DRIVE.TYPE:
                  ;15689 ;
                  ;15690 ;        BEN0/R80,               ;WHICH TYPE OF DRIVE WAS SELECTED?
                  ;15691 ;        NAD/RL02.IFORK
                  ;15692 ;
                  ;15693 ;------------------------------------------------------------------
                  ;15694 ;=0
                  ;15695 ;
                  ;15696 ;RL02.IFORK:
                  ;15697 ;
                  ;15698 ;        UCMD/SET.FIFOA,
                  ;15699 ;        FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
                  ;15700 ;        BEN0/F0,                ;8 WAY INSTRUCTION FORK.
                  ;15701 ;        BEN1/F1,                ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                  ;15702 ;        BEN2/F2,
                  ;15703 ;        NAD/NOP.CMD
                  ;15704 ;
                  ;15705 ;------------------------------------------------------------------
                  ;15706 ;=010
                  ;15707 ;
                  ;15708 ;GET.STAT.CMD:
                  ;15709 ;
                  ;15710 ;        NAD/SETUP.FIFO2         ;START OF THE RL02 GET STATUS COMMAND.
                  ;15711 ;
                  ;15712 ;------------------------------------------------------------------
                  ;15713 ;
```

```
;SETUP.FIFO2:
;
;       UCMD/SET.FIFOA, ;SELECT FIFOA AND CLEAR ADDRESS CNTR.
;       CLR.FIFO.ADR
;
;------------------------------------------------------------------------
;
;CHGCLK2:
;
;       ENB.CLR.BC/ON,          ;CLEAR BC SO THAT DARO IS SELECTED
;       ENB.SYS.CLK             ;USE RL02 WRITE CLOCK
;
;------------------------------------------------------------------------
;
;SETUP.DRV.CMD2:
;
;       ENB.CLR.BC/ON,          ;PREPARE TO SEND COMMAND.
;       LOAD.LOOP.CNTR.[0F0]
;
;------------------------------------------------------------------------
;=0
;
;SEND.DRV.CMD2:
;
;       ENB.DRV.CMD/ON,         ;SHIFT OUT GET STATUS COMMAND
;       INCR.LPCNTR/ON,
;       BENO/CNTR.OVFLW,
;       NAD/SEND.DRV.CMD2
;
;------------------------------------------------------------------------
;
;CHGCLK2A:
;
;       ENB.STATUS.CLK,         ;RUN OFF THE STATUS CLOCK
;       LOAD.LOOP.CNTR.[0FA]    ; Fix for new version
;
;------------------------------------------------------------------------
;=0
;
;READ.STATUS2:
;
;       INCR.LPCNTR/ON,         ;WHEN SEQ CLK STARTS, THERE WILL ALREADY
;       BENO/CNTR.OVFLW,        ;BE 3 STATUS BITS IN THE DSR.
;       NAD/READ.STATUS2
;
;------------------------------------------------------------------------
;=1
;
;STORE.HIBYTE2:
;
;       ENB.DSR/ON,             ;8TH STATUS BIT
;       WRT.FIFO/ON,            ;LOAD BYTE INTO FIFO
;       LOAD.LOOP.CNTR.[0F9],
;       INCR.FIFO.CNTR/ON
;
```

```
;-----------------------------------------------------------------
;=0
;
;READ.STATUS2A:
;
;        INCR.LPCNTR/ON,          ;SHIFT IN 2ND BYTE
;        BENO/CNTR.OVFLW,
;        NAD/READ.STATUS2A
;
;-----------------------------------------------------------------
;=1
;
;STORE.LOBYTE2:
;
;        ENB.DSR/ON,              ;LAST STATUS BIT
;        WRT.FIFO/ON              ;LOAD BYTE INTO FIFO
;
;-----------------------------------------------------------------
;
;END.STAT.CMD2:
;
;        ENB.CPU.CLK,             ;BACK TO THE NEBULA CLOCK.
;        CLR.FIFO.ADR,            ;CLEAR THE FIFO ADDRESS COUNTER
;        NAD/XFER.DONE4
;
;-----------------------------------------------------------------
;
;XFER.DONE4:
;
;        UCMD/SET.CRDY,           ;SIGNAL END OF TRANSFER.
;        FUNC/SET.INT,
;        NAD/SET.INT.AND.IDLE
;
;-----------------------------------------------------------------
;=111111110
;
;SET.INT.AND.IDLE:
;
;        FUNC/SET.INT,            ; The purpose of this state is to give CRDY one
;        NAD/IDLE                 ; state to reset before asserting interrupt.
;                                 ; CRDY is now used in the INT equation.
;
;-----------------------------------------------------------------
;=1
;
;RTN.TO.IDLE:
;IDLE:
;
;        BENO/CRDY.L,             ;LOOK FOR GO (CRDY L)
;        NAD/SETUP.DRV.CHK
;
;-----------------------------------------------------------------
;=1
;
;TEST.OR.RUN:
```

```
                   ;15824 ;
                   ;15825 ;          BEN2/MAINT,                ;IS THAT YOU, VICKIE?
                   ;15826 ;          NAD/DRIVE.TYPE
                   ;15827 ;
                   ;15828 ;------------------------------------------------------------------
                   ;15829 ;=0**
                   ;15830 ;
                   ;15831 ;DRIVE.TYPE:
                   ;15832 ;
                   ;15833 ;          BEN0/R80,                  ;WHICH TYPE OF DRIVE WAS SELECTED?
                   ;15834 ;          NAD/RL02.IFORK
                   ;15835 ;
                   ;15836 ;------------------------------------------------------------------
                   ;15837 ;=0
                   ;15838 ;
                   ;15839 ;RL02.IFORK:
                   ;15840 ;
                   ;15841 ;          UCMD/SET.FIFOA,
                   ;15842 ;          FUNC/CLR.ECC.ERROR,        ; Enable clear of ECC/CRC errors for new function
                   ;15843 ;          BEN0/F0,                   ;8 WAY INSTRUCTION FORK.
                   ;15844 ;          BEN1/F1,                   ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                   ;15845 ;          BEN2/F2,
                   ;15846 ;          NAD/NOP.CMD
                   ;15847 ;
                   ;15848 ;------------------------------------------------------------------
                   ;15849 ;=000
                   ;15850 ; This was formerly patched to enter diskless loop
                   ;15851 ; This version makes it a true NOP
                   ;15852 ;
                   ;15853 ;NOP.CMD:
                   ;15854 ;
                   ;15855 ;          UCMD/SET.CRDY,             ;SIGNAL END OF FUNCTION.
                   ;15856 ;          FUNC/SET.INT,
                   ;15857 ;          NAD/SET.INT.AND.IDLE               .
                   ;15858 ;
                   ;15859 ;------------------------------------------------------------------
                   ;15860
                   ;15861
                   ;15862 WAIT.IDC.TF.1:
U 0CBB, 88,700C    ;15863          JSR [WAIT.CSR.CRDY]                        ;check if CRDY is set
U 0CBC, 88,CBE4    ;15864          JMP [IDC.DONE.TF.1]                        ;yes, CRDY is set now
U 0CBD, 88,CBB4    ;15865          JMP [WAIT.IDC.TF.1]                        ;no, wait some more
                   ;15866
                   ;15867 ;Read the current drive status from FIFO A
                   ;15868
                   ;15869 IDC.DONE.TF.1:
U 0CBE, 17,D815    ;15870          PORT.CONTROL [SEL.FIFO.A]                  ;select FIFO A
U 0CBF, 17,C015    ;15871          PORT.CONTROL [CLEAR.FIFO.CNTR]             ;clear the FIFO A adr cntr.
U 0CC0, 90,9015    ;15872          MISC [SET.PORT.I.O]                        ;select IDC
U 0CC1, 97,8C15    ;15873          PORT.READ PORT.ADRS[DATA.WORD]             ;read status from FIFO A
U 0CC2, 38,3215    ;15874          MOV PORT TO LS[PORT0]
U 0CC3, 37,3215    ;15875          MOV LS[PORT0] TO WR[0]
U 0CC4, 37,9695    ;15876          MOV LS[#7] TO WR[1]                        ;mask for state bits
U 0CC5, 2F,C215    ;15877          AND WR[1] TO WR[0]
                   ;15878
```

```
                 ;Set uo parameters for error reporting routine.

U 0CC6, B6,4295          MOV LS[#2] TO WR[1]
U 0CC7, 3E,8295          MOV WR[1] TO LS[ERROR.NUMBER]           ;set up error number = 2 in LS
U 0CC8, 37,9A95          MOV LS[#FFFFFFF8] TO WR[1]              ;error mask value
U 0CC9, BE,8A95          MOV WR[1] TO LS[ERROR.MASK]
U 0CCA, 36,4A95          MOV LS[IDC] TO WR[1]                    ;store module code in LS to
U 0CCB, 47,4C95          BIS LS[RL02] TO WR[1]                   ; to indicate IDC and RL02
U 0CCC, BE,8C95          MOV WR[1] TO LS[MODULE.NUM]             ; modules
U 0CCD, B7,9295          MOV LS[#5] TO WR[1]                     ;expected state bits
U 0CCE, B6,4915          MOV LS[BIT4] TO WR[2]                   ;no, set bit4 to CRD micmon
U 0CCF, BE,8115          MOV WR[2] TO LS[CONTROL.STATUS]         ; for printing of error msg.

                 ;Analyze only the state bits <2:0> of the RL02 status word.

U 0CD0, B7,9295          MOV LS[#5] TO WR[1]                     ;expected lock on state = 5
                         CMP WR[0] WITH WR[1],                   ;is state "IN LOCK ON" ?
U 0CD1, 26,40B5           DT(LONG)&SET.ALU.CC
U 0CD2, 88,CE39          JMP.IF[EQL] TO [TEST.TF.3]              ;yes, check remaining status

U 0CD3, 2F,8295          CLR WR[1]
                         CMP WR[0] WITH WR[1],                   ;is state "LOAD CARTRIDGE" ?
U 0CD4, 26,40B5           DT(LONG)&SET.ALU.CC
U 0CD5, 08,CDE9          JMP.IF[EQL] TO [TF.WARN.ER.1]           ;issue warning msg. to console

U 0CD6, 37,8E95          MOV LS[WAIT.CNTR] TO WR[1]              ;inspect the wait loop counter
U 0CD7, 37,9115          MOV LS[WAIT.TIME1] TO WR[2]             ;get the wait time
                         CMP WR[1] WITH WR[2],                   ;exceed wait time limit ?
U 0CD8, A6,4335           DT(LONG)&SET.ALU.CC
U 0CD9, B7,9295          MOV LS[#5] TO WR[1]                     ;restore exp. data in WR[1]
U 0CDA, 88,CE09          JMP.IF[EQL] TO [TF.ER.1]                ;yes, go print the error

U 0CDB, 20,4295          INC WR[1]                               ;no, increment the
U 0CDC, BF,8E95          MOV WR[1] TO LS[WAIT.CNTR]              ; wait counter by 1
U 0CDD, 08,CB54          JMP [TEST.TF.2]                         ;give RL02 more time to settle

                 TF.WARN.ER.1:
U 0CDE, 10,E015          MISC [SET.CP.ATTN]                      ;report the warning

                 WAIT.TF.2:
U 0CDF, 08,CDF4          JMP [WAIT.TF.2]                         ;wait for console to respond

                 TF.ER.1:
U 0CE0, 08,693C          JSR [CHECK.RESULT]                      ;enter here to get a standard
                                                                 ; error report for reg. micmon
U 0CE1, 08,CB34          JMP [GET.STATUS.1]                      ;loop on err if err bit is set
U 0CE2, 08,CE34          JMP [TEST.TF.3]                         ;no error found, proceed.

                 ;Analyze the remaining status bits with bits <13, 7 & 6> masked out.

                 TEST.TF.3:
U 0CE3, B6,C095          MOV LS[#3] TO WR[1]
U 0CE4, 3E,8295          MOV WR[1] TO LS[ERROR.NUMBER]           ;set up error number = 3 in LS
U 0CE5, 37,A095          MOV LS[#FFFF20C7] TO WR[1]              ;error mask value
U 0CE6, BE,8A95          MOV WR[1] TO LS[ERROR.MASK]
```

```
U 0CE7, 36,4A95  ;15934         MOV LS[IDC] TO WR[1]                    ;store module code in LS to
U 0CE8, 47,4C95  ;15935         BIS LS[RL02] TO WR[1]                   ; to indicate IDC and RL02
U 0CE9, BE,8C95  ;15936         MOV WR[1] TO LS[MODULE.NUM]             ; modules
U 0CEA, B7,A895  ;15937         MOV LS[#00000018] TO WR[1]              ;expected status bits
U 0CEB, 37,3215  ;15938         MOV LS[PORT0] TO WR[0]                  ;received status
U 0CEC, 08,693C  ;15939         JSR [CHECK.RESULT]                      ;good RL02 status to proceed ?
U 0CED, 08,CE34  ;15940         JMP [TEST.TF.3]                         ;no, loop on error
U 0CEE, 08,CEF4  ;15941         JMP [TEST.TF.4]                         ;yes, proceed
                 ;15942
                 ;15943 ;Now the status is ok, proceed by locating the heads on the disk.
                 ;15944
                 ;15945 TEST.TF.4:
U 0CEF, B7,0615  ;15946         MOV LS[READ.HEADER] TO WR[0]            ;get the read_header command
U 0CF0, 47,5015  ;15947         BIS LS[SETCSR.DS1] TO WR[0]             ;select drive 1
U 0CF1, 17,A015  ;15948         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;write the CSR
                 ;15949                                                 ;MAINT=0,CRDY=0,F<2:0>=100
                 ;15950
                 ;15951 ;*** RL02 READ HEADER IDC FUNCTION CODES : ***
                 ;15952
                 ;15953 ;---------------------------------------------------------------------
                 ;15954 ;=1
                 ;15955 ;
                 ;15956 ;TEST.OR.RUN:
                 ;15957 ;
                 ;15958 ;       BEN2/MAINT,             ;IS THAT YOU, VICKIE?
                 ;15959 ;       NAD/DRIVE.TYPE
                 ;15960 ;
                 ;15961 ;---------------------------------------------------------------------
                 ;15962 ;=0**
                 ;15963 ;
                 ;15964 ;DRIVE.TYPE:
                 ;15965 ;
                 ;15966 ;       BEN0/R80,               ;WHICH TYPE OF DRIVE WAS SELECTED?
                 ;15967 ;       NAD/RL02.IFORK
                 ;15968 ;
                 ;15969 ;---------------------------------------------------------------------
                 ;15970 ;=0
                 ;15971 ;
                 ;15972 ;RL02.IFORK:
                 ;15973 ;
                 ;15974 ;       UCMD/SET.FIFOA,
                 ;15975 ;       FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
                 ;15976 ;       BEN0/F0,                ;8 WAY INSTRUCTION FORK.
                 ;15977 ;       BEN1/F1,                ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                 ;15978 ;       BEN2/F2,
                 ;15979 ;       NAD/NOP.CMD
                 ;15980 ;
                 ;15981 ;---------------------------------------------------------------------
                 ;15982 ;=100
                 ;15983 ;
                 ;15984 ;RD.HDR.CMD:
                 ;15985 ;
                 ;15986 ;       NAD/WAIT.DRDY4          ;START OF RL02 READ HEADER COMMAND.
                 ;15987 ;
                 ;15988 ;---------------------------------------------------------------------
```

```
;15989 ;=0
;15990 ;
;15991 ;WAIT.DRDY4:
;15992 ;
;15993 ;        BEN0/DRIVE.RDY,             ;WAIT FOR DRIVE READY
;15994 ;        NAD/WAIT.DRDY4
;15995 ;
;15996 ;-------------------------------------------------------------------
;15997 ;=1
;15998 ;
;15999 ;CHGCLK4:
;16000 ;
;16001 ;        ENB.SYS.CLK                 ;USE RL02 WRITE CLOCK
;16002 ;
;16003 ;-------------------------------------------------------------------
;16004 ;=0*
;16005 ;
;16006 ;WAIT.SEC.PLS4:
;16007 ;
;16008 ;        BEN1/SECTOR.PLS,            ;WAIT FOR LEADING EDGE OF SECTOR PULSE
;16009 ;        NAD/WAIT.SEC.PLS4
;16010 ;
;16011 ;-------------------------------------------------------------------
;16012 ;=1*
;16013 ;
;16014 ;SEC.PLS.FND4:
;16015 ;
;16016 ;        CLR.FIFO.ADR,
;16017 ;        NAD/WAIT.END.SEC.PLS4       ;SECTOR PULSE HAS BEEN FOUND
;16018 ;
;16019 ;-------------------------------------------------------------------
;16020 ;=1*
;16021 ;
;16022 ;WAIT.END.SEC.PLS4:
;16023 ;
;16024 ;        BEN1/SECTOR.PLS,            ;WAIT FOR END OF SECTOR PULSE
;16025 ;        NAD/WAIT.END.SEC.PLS4
;16026 ;
;16027 ;-------------------------------------------------------------------
;16028 ;=1
;16029 ;
;16030 ;SETUP.RD.HDR4:
;16031 ;
;16032 ;        LOOP.LOCK/ON,               ;SETUP INITIAL CONDITIONS
;16033 ;        FUNC/CLR.DSR,               ;FOR READING HEADER.
;16034 ;        LOAD.LOOP.CNTR.[0F9]
;16035 ;
;16036 ;-------------------------------------------------------------------
;16037 ;
;16038 ;CHGCLK4A:
;16039 ;
;16040 ;        ENB.DSK.CLK,                ;THE MICROSEQUENCER WILL STALL
;16041 ;        LOOP.LOCK/ON,               ;UNTIL SYNC SEEN IS SET (SYNC BIT FOUND)
;16042 ;        GEN.SYNC.[80]
;16043 ;
```

```
;16044 ;-----------------------------------------------------------------------
;16045 ;=0
;16046 ;
;16047 ;READ.HDR.BYTE4:
;16048 ;
;16049 ;        LOOP.LOCK/ON,             ;READ FIRST HEADER BYTE
;16050 ;        INCR.LPCNTR/ON,
;16051 ;        BENO/CNTR.OVFLW,
;16052 ;        NAD/READ.HDR.BYTE4
;16053 ;
;16054 ;-----------------------------------------------------------------------
;16055 ;=1
;16056 ;
;16057 ;STORE.HDR.BYTE4:
;16058 ;
;16059 ;        LOOP.LOCK/ON,             ;STORE FIRST HDR BYTE IN FIFO
;16060 ;        ENB.DSR/ON,
;16061 ;        WRT.FIFO/ON,
;16062 ;        INCR.FIFO.CNTR/ON,
;16063 ;        LOAD.LOOP.CNTR.[0F9]
;16064 ;
;16065 ;-----------------------------------------------------------------------
;16066 ;=0
;16067 ;
;16068 ;READ.HDR.BYTE4A:
;16069 ;
;16070 ;        LOOP.LOCK/ON,             ;READ SECOND HDR BYTE
;16071 ;        INCR.LPCNTR/ON,
;16072 ;        BENO/CNTR.OVFLW,
;16073 ;        NAD/READ.HDR.BYTE4A
;16074 ;
;16075 ;-----------------------------------------------------------------------
;16076 ;=1
;16077 ;
;16078 ;STORE.HDR.BYTE4A:
;16079 ;
;16080 ;        LOOP.LOCK/ON,             ;STORE SECOND HDR BYTE IN FIFO
;16081 ;        ENB.DSR/ON,
;16082 ;        WRT.FIFO/ON,
;16083 ;        INCR.FIFO.CNTR/ON,
;16084 ;        LOAD.LOOP.CNTR.[0F2]
;16085 ;
;16086 ;-----------------------------------------------------------------------
;16087 ;=0
;16088 ;
;16089 ;SKIP.HDR.WORD4:
;16090 ;
;16091 ;        LOOP.LOCK/ON,             ;SKIP THIRD & FOURTH HEADER BYTE
;16092 ;        INCR.LPCNTR/ON,
;16093 ;        BENO/CNTR.OVFLW,
;16094 ;        NAD/SKIP.HDR.WORD4
;16095 ;
;16096 ;-----------------------------------------------------------------------
;16097 ;=1
;16098 ;
```

```
;16099 ;SKIP.HDR.WORD4A:
;16100 ;
;16101 ;        LOOP.LOCK/ON,              ;SETUP FOR HDR CRC BYTES
;16102 ;        LOAD.LOOP.CNTR.[OF8]
;16103 ;
;16104 ;--------------------------------------------------------------------
;16105 ;=0
;16106 ;
;16107 ;READ.CRC.BYTE4:
;16108 ;
;16109 ;        LOOP.LOCK/ON,              ;READ FIRST HEADER CRC BYTE
;16110 ;        INCR.LPCNTR/ON,
;16111 ;        BENO/CNTR.OVFLW,
;16112 ;        NAD/READ.CRC.BYTE4
;16113 ;
;16114 ;--------------------------------------------------------------------
;16115 ;=1
;16116 ;
;16117 ;STORE.CRC.BYTE4:
;16118 ;
;16119 ;        LOOP.LOCK/ON,              ;STORE FIRST HDR CRC BYTE IN FIFO
;16120 ;        ENB.DSR/ON,
;16121 ;        WRT.FIFO/ON,
;16122 ;        INCR.FIFO.CNTR/ON,
;16123 ;        LOAD.LOOP.CNTR.[OF9]
;16124 ;
;16125 ;--------------------------------------------------------------------
;16126 ;=0
;16127 ;
;16128 ;READ.CRC.BYTE4A:
;16129 ;
;16130 ;        LOOP.LOCK/ON,              ;READ SECOND HDR CRC BYTE
;16131 ;        INCR.LPCNTR/ON,
;16132 ;        BENO/CNTR.OVFLW,
;16133 ;        NAD/READ.CRC.BYTE4A
;16134 ;
;16135 ;--------------------------------------------------------------------
;16136 ;=1
;16137 ;
;16138 ;STORE.CRC.BYTE4A:
;16139 ;
;16140 ;        LOOP.LOCK/ON,              ;STORE SECOND HDR CRC BYTE IN FIFO
;16141 ;        ENB.DSR/ON,
;16142 ;        INCR.FIFO.CNTR/ON,
;16143 ;        WRT.FIFO/ON
;16144 ;
;16145 ;--------------------------------------------------------------------
;16146 ;
;16147 ;CLEAR.FIFO.ADRS4:
;16148 ;
;16149 ;        LOOP.LOCK/ON,
;16150 ;        CLR.FIFO.ADR,              ;CLEAR THE FIFO ADRS COUNTER.
;16151 ;        NAD/CHGCLK4B
;16152 ;
;16153 ;--------------------------------------------------------------------
```

```
;
;CHGCLK4B:
;
;       LOOP.LOCK/ON,           ;SWITCH OVER TO CPU CLOCK
;       ENB.CPU.CLK             ;DURING POSTAMBLE.
;
;-----------------------------------------------------------------------
;
;XFER.DONE4:
;
;       UCMD/SET.CRDY,          ;SIGNAL END OF TRANSFER.
;       FUNC/SET.INT,
;       NAD/SET.INT.AND.IDLE
;
;-----------------------------------------------------------------------
;=111111110
;
;SET.INT.AND.IDLE:
;
;       FUNC/SET.INT,           ; The purpose of this state is to give CRDY one
;       NAD/IDLE                ; state to reset before asserting interrupt.
;                               ; CRDY is now used in the INT equation.
;
;-----------------------------------------------------------------------
;=1
;
;RTN.TO.IDLE:
;IDLE:
;
;       BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
;       NAD/SETUP.DRV.CHK
;
;-----------------------------------------------------------------------
;=1
;
;TEST.OR.RUN:
;
;       BEN2/MAINT,             ;IS THAT YOU, VICKIE?
;       NAD/DRIVE.TYPE
;
;-----------------------------------------------------------------------
;=0**
;
;DRIVE.TYPE:
;
;       BENO/R80,               ;WHICH TYPE OF DRIVE WAS SELECTED?
;       NAD/RL02.IFORK
;
;-----------------------------------------------------------------------
;=0
;
;RL02.IFORK:
;
;       UCMD/SET.FIFOA,
;       FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
```

```
                ;16209 ;        BEN0/F0,                ;8 WAY INSTRUCTION FORK.
                ;16210 ;        BEN1/F1,                ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                ;16211 ;        BEN2/F2,
                ;16212 ;        NAD/NOP.CMD
                ;16213 ;
                ;16214 ;--------------------------------------------------------------------
                ;16215 ;=000
                ;16216 ; This was formerly patched to enter diskless loop
                ;16217 ; This version makes it a true NOP
                ;16218 ;
                ;16219 ;NOP.CMD:
                ;16220 ;
                ;16221 ;        UCMD/SET.CRDY,          ;SIGNAL END OF FUNCTION.
                ;16222 ;        FUNC/SET.INT,
                ;16223 ;        NAD/SET.INT.AND.IDLE
                ;16224 ;
                ;16225 ;--------------------------------------------------------------------
                ;16226
                ;16227
                ;16228
                ;16229
                ;16230 WAIT.IDC.TF.4:
U 0CF2, 88,700C ;16231          JSR [WAIT.CSR.CRDY]                     ;check if CRDY is set
U 0CF3, 88,CF54 ;16232          JMP [TEST.TF.5]                         ;yes, CRDY is set now
U 0CF4, 08,CF24 ;16233          JMP [WAIT.IDC.TF.4]                     ;no, wait some more
                ;16234
                ;16235 ;Use a seek function to position the heads to the first cylinder of the disk.
                ;16236
                ;16237 TEST.TF.5:
U 0CF5, B6,9C95 ;16238          MOV LS[ZERO] TO WR[1]                   ;fetch the desired seek cyl.
U 0CF6, BF,8495 ;16239          MOV WR[1] TO LS[SEEK.CYL]
U 0CF7, 88,72EC ;16240          JSR [SHIFT.HDR]                         ;shift header into DAR format
U 0CF8, 08,745C ;16241          JSR [SEEK.DAR]                          ;set up DAR seek format
U 0CF9, B6,0215 ;16242          MOV LS[SAV.WR0] TO WR[0]
U 0CFA, 97,A415 ;16243          PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ;right DAR seek format
                ;16244
U 0CFB, 37,0415 ;16245          MOV LS[SEEK] TO WR[0]                   ;get the seek command
U 0CFC, 47,5015 ;16246          BIS LS[SETCSR.DS1] TO WR[0]             ;select drive 1
U 0CFD, 17,A015 ;16247          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;bring the head home
                ;16248
                ;16249 ;*** RL02 SEEK IDC FUNCTION CODES : ***
                ;16250
                ;16251 ;--------------------------------------------------------------------
                ;16252 ;=1
                ;16253 ;
                ;16254 ;TEST.OR.RUN:
                ;16255 ;
                ;16256 ;        BEN2/MAINT,             ;IS THAT YOU, VICKIE?
                ;16257 ;        NAD/DRIVE.TYPE
                ;16258 ;
                ;16259 ;--------------------------------------------------------------------
                ;16260 ;=0**
                ;16261 ;
                ;16262 ;DRIVE.TYPE:
                ;16263 ;
```

```
;16264 ;        BENO/R80,                ;WHICH TYPE OF DRIVE WAS SELECTED?
;16265 ;        NAD/RL02.IFORK
;16266 ;
;16267 ;---------------------------------------------------------------------
;16268 ;=0
;16269 ;
;16270 ;RL02.IFORK:
;16271 ;
;16272 ;        UCMD/SET.FIFOA,
;16273 ;        FUNC/CLR.ECC.ERROR,      ; Enable clear of ECC/CRC errors for new function
;16274 ;        BENO/FO,                 ;8 WAY INSTRUCTION FORK.
;16275 ;        BEN1/F1,                 ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
;16276 ;        BEN2/F2,
;16277 ;        NAD/NOP.CMD
;16278 ;
;16279 ;---------------------------------------------------------------------
;16280 ;=011
;16281 ;
;16282 ;SEEK.CMD:
;16283 ;
;16284 ;        NAD/WAIT.DRDY3           ;START OF THE RL02 SEEK COMMAND.
;16285 ;
;16286 ;---------------------------------------------------------------------
;16287 ;=0
;16288 ;
;16289 ;WAIT.DRDY3:
;16290 ;
;16291 ;        BENO/DRIVE.RDY,          ;WAIT FOR DRIVE READY
;16292 ;        NAD/WAIT.DRDY3
;16293 ;
;16294 ;---------------------------------------------------------------------
;16295 ;=1
;16296 ;
;16297 ;CHGCLK3:
;16298 ;
;16299 ;        ENB.SYS.CLK              ;USE RL02 WRITE CLOCK
;16300 ;
;16301 ;---------------------------------------------------------------------
;16302 ;=0*
;16303 ;
;16304 ;WAIT.SEC.PLS3:
;16305 ;
;16306 ;        BEN1/SECTOR.PLS,         ;WAIT FOR LEADING EDGE OF SECTOR PULSE
;16307 ;        NAD/WAIT.SEC.PLS3
;16308 ;
;16309 ;---------------------------------------------------------------------
;16310 ;=1*
;16311 ;
;16312 ;SEC.PLS.FND3:
;16313 ;
;16314 ;        NAD/WAIT.END.SEC.PLS3    ;SECTOR PULSE HAS BEEN FOUND
;16315 ;
;16316 ;---------------------------------------------------------------------
;16317 ;=0*
;16318 ;
```

```
;END.SEC.PLS3:
;
;	LOAD.LOOP.CNTR.[2E],	;PREPARE FOR WAIT LOOP
;	NAD/WAIT.50.US3
;
;-----------------------------------------------------------------------
;=1*
;
;WAIT.END.SEC.PLS3:
;
;	BEN1/SECTOR.PLS,	;WAIT FOR END OF SECTOR PULSE
;	NAD/WAIT.END.SEC.PLS3
;
;-----------------------------------------------------------------------
;=0
;
;WAIT.50.US3:
;
;	INCR.LPCNTR/ON,		;WAIT 50 US BEFORE SENDING DRIVE CMD.
;	ENB.CLR.BC/ON,		;CLEAR BC SO THAT DARO IS SELECTED
;	BENO/CNTR.OVFLW,
;	NAD/WAIT.50.US3
;
;-----------------------------------------------------------------------
;=1
;
;SETUP.DRV.CMD3:
;
;	ENB.CLR.BC/ON,		;PREPARE TO SEND COMMAND.
;	LOAD.LOOP.CNTR.[0F0]
;
;-----------------------------------------------------------------------
;=0
;
;SEND.DRV.CMD3:
;
;	ENB.DRV.CMD/ON,		;SHIFT OUT DRIVE COMMAND
;	INCR.LPCNTR/ON,
;	BENO/CNTR.OVFLW,
;	NAD/SEND.DRV.CMD3
;
;-----------------------------------------------------------------------
;=1
;
;DRV.CMD.DONE3:
;
;	ENB.CPU.CLK,		;SWITCH BACK TO IDLE CLOCK
;	FUNC/CLR.ONLINE,	; Avoid setting attn bit on -DRDY
;	NAD/XFER.DONE4
;
;-----------------------------------------------------------------------
;
;XFER.DONE4:
;
;	UCMD/SET.CRDY,		;SIGNAL END OF TRANSFER.
```

```
;       FUNC/SET.INT,
;       NAD/SET.INT.AND.IDLE
;
;-----------------------------------------------------------------
;=111111110
;
;SET.INT.AND.IDLE:
;
;       FUNC/SET.INT,           ; The purpose of this state is to give CRDY one
;       NAD/IDLE                ; state to reset before asserting interrupt.
;                               ; CRDY is now used in the INT equation.
;
;-----------------------------------------------------------------
;=1
;
;RTN.TO.IDLE:
;IDLE:
;
;       BEN0/CRDY.L,            ;LOOK FOR GO (CRDY L)
;       NAD/SETUP.DRV.CHK
;
;-----------------------------------------------------------------
;=1
;
;TEST.OR.RUN:
;
;       BEN2/MAINT,             ;IS THAT YOU, VICKIE?
;       NAD/DRIVE.TYPE
;
;-----------------------------------------------------------------
;=0**
;
;DRIVE.TYPE:
;
;       BEN0/R80,               ;WHICH TYPE OF DRIVE WAS SELECTED?
;       NAD/RL02.IFORK
;
;-----------------------------------------------------------------
;=0
;
;RL02.IFORK:
;
;       UCMD/SET.FIFOA,
;       FUNC/CLR.ECC.ERROR,     ; Enable clear of ECC/CRC errors for new function
;       BEN0/F0,                ;8 WAY INSTRUCTION FORK.
;       BEN1/F1,                ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
;       BEN2/F2,
;       NAD/NOP.CMD
;
;-----------------------------------------------------------------
;=000
; This was formerly patched to enter diskless loop
; This version makes it a true NOP
;
;NOP.CMD:
```

```
                    ;16429 ;
                    ;16430 ;        UCMD/SET.CRDY,          ;SIGNAL END OF FUNCTION.
                    ;16431 ;        FUNC/SET.INT,
                    ;16432 ;        NAD/SET.INT.AND.IDLE
                    ;16433 ;
                    ;16434 ;-------------------------------------------------------------------
                    ;16435
U 0CFE, DB,0015     ;16436          NOP                                     ;added to avoid validity
U 0CFF, DB,0015     ;16437          NOP                                     ; failure in the assembly
U 0D00, DB,0015     ;16438          NOP                                     ; of this program
                    ;16439
                    ;16440 WAIT.IDC.TF.5:
U 0D01, 88,700C     ;16441          JSR [WAIT.CSR.CRDY]                     ;check if CRDY is set
U 0D02, 88,D044     ;16442          JMP [TEST.TF.6]                         ;yes, CRDY is set now
U 0D03, 88,D014     ;16443          JMP [WAIT.IDC.TF.5]                     ;no, wait some more
                    ;16444
                    ;16445 TEST.TF.6:
U 0D04, B7,0615     ;16446          MOV LS[READ.HEADER] TO WR[0]            ;get the read_header command
U 0D05, 47,5015     ;16447          BIS LS[SETCSR.DS1] TO WR[0]                     ;select drive 1
U 0D06, 17,A015     ;16448          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;write the CSR
                    ;16449                                                  ;MAINT=0,CRDY=0,F<2:0>=100
                    ;16450
                    ;16451          ;*************************************************************
                    ;16452          ;* Refer to the RL02 read header IDC function codes as listed above *
                    ;16453          ;*************************************************************
                    ;16454
                    ;16455 WAIT.IDC.TF.6:
U 0D07, 88,700C     ;16456          JSR [WAIT.CSR.CRDY]                     ;check if CRDY is set
U 0D08, 08,D0A4     ;16457          JMP [FIND.HEAD.TF.2]                    ;xfer req set
U 0D09, 88,D074     ;16458          JMP [WAIT.IDC.TF.6]                     ;wait some more
                    ;16459
                    ;16460 ;Verify if the last seek command did bring the heads to the first cylinder.
                    ;16461
                    ;16462 FIND.HEAD.TF.2:
U 0D0A, 88,72EC     ;16463          JSR [SHIFT.HDR]                         ;shift header to DAR format
U 0D0B, B6,4495     ;16464          MOV LS[#4] TO WR[1]
U 0D0C, 3E,8295     ;16465          MOV WR[1] TO LS[ERROR.NUMBER]           ;update the error number = 4
U 0D0D, B7,A295     ;16466          MOV LS[#FFFF007F] TO WR[1]              ;error mask for
U 0D0E, BE,8A95     ;16467          MOV WR[1] TO LS[ERROR.MASK]             ; the RL02 cyl. address
U 0D0F, 36,4A95     ;16468          MOV LS[IDC] TO WR[1]                    ;identify the IDC and RL02
U 0D10, 47,4C95     ;16469          BIS LS[RL02] TO WR[1]                   ; modules at fault
U 0D11, BE,8C95     ;16470          MOV WR[1] TO LS[MODULE.NUM]             ; for error printout
U 0D12, B6,0215     ;16471          MOV LS[SAV.WR0] TO WR[0]                ;load received cylinder adr.
U 0D13, B6,9C95     ;16472          MOV LS[ZERO] TO WR[1]                   ;expected data for cyl. adr.
U 0D14, 08,693C     ;16473          JSR [CHECK.RESULT]                      ;check for cyl. 0
U 0D15, 08,D0A4     ;16474          JMP [FIND.HEAD.TF.2]                    ;loop on err if err bit is set
                    ;16475
                    ;16476 ; The following section reads the lower 256 bytes of the homeblock into FIFO A
                    ;16477 ; initially from the 3rd sector of track 0 and cylinder 0. it then determines
                    ;16478 ; if it is a valid homeblock by checking for non zero content of the 2nd and
                    ;16479 ; 3rd longwords (alternate home block - H.AHLB & backup index file header -
                    ;16480 ; H.IHLB as in Files-11 On-disk Structure Specification by Andrew C. Goldstein;
                    ;16481 ; document # 130-958-032-03; dated 1-APR-1978).
                    ;16482
                    ;16483      ; Initialize counters for cylinder and sector.
```

```
                   ;16484
U 0D16, 2F,8295    ;16485            CLR WR[1]
U 0D17, BF,8895    ;16486            MOV WR[1] TO LS[CYL.CNTR]                ;init. cyl. to 0
U 0D18, 3F,8A95    ;16487            MOV WR[1] TO LS[DAR.SEEK]                ;init. DAR seek format
U 0D19, B6,4295    ;16488            MOV LS[#2] TO WR[1]                      ;init. sector no. to 3
                   ;16489
                   ;16490 FIND.HBLK:
U 0D1A, BE,0495    ;16491            MOV WR[1] TO LS[SAV.WR1]                 ;save the current sector
U 0D1B, 17,A495    ;16492            PORT.WRITE PORT.ADRS[DAR] WITH WR[1]     ;get to the rite sector
U 0D1C, B7,0A15    ;16493            MOV LS[READ.DATA] TO WR[0]               ;setup a read data command
U 0D1D, 47,5015    ;16494            BIS LS[SETCSR.DS1] TO WR[0]              ;select drive 1
U 0D1E, 17,A015    ;16495            PORT.WRITE PORT.ADRS[CSR] WITH WR[0]     ;write the CSR
                   ;16496
                   ;16497 ;*** RL02 READ DATA IDC FUNCTION CODES : ***
                   ;16498
                   ;16499 ;-----------------------------------------------------------------------
                   ;16500 ;=1
                   ;16501 ;
                   ;16502 ;TEST.OR.RUN:
                   ;16503 ;
                   ;16504 ;          BEN2/MAINT,               ;IS THAT YOU, VICKIE?
                   ;16505 ;          NAD/DRIVE.TYPE
                   ;16506 ;
                   ;16507 ;-----------------------------------------------------------------------
                   ;16508 ;=0**
                   ;16509 ;
                   ;16510 ;DRIVE.TYPE:
                   ;16511 ;
                   ;16512 ;          BEN0/R80,                 ;WHICH TYPE OF DRIVE WAS SELECTED?
                   ;16513 ;          NAD/RL02.IFORK
                   ;16514 ;
                   ;16515 ;-----------------------------------------------------------------------
                   ;16516 ;=0
                   ;16517 ;
                   ;16518 ;RL02.IFORK:
                   ;16519 ;
                   ;16520 ;          UCMD/SET.FIFOA,
                   ;16521 ;          FUNC/CLR.ECC.ERROR,       ; Enable clear of ECC/CRC errors for new function
                   ;16522 ;          BEN0/F0,                  ;8 WAY INSTRUCTION FORK.
                   ;16523 ;          BEN1/F1,                  ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                   ;16524 ;          BEN2/F2,
                   ;16525 ;          NAD/NOP.CMD
                   ;16526 ;
                   ;16527 ;-----------------------------------------------------------------------
                   ;16528 ;=110
                   ;16529 ;
                   ;16530 ;RD.DAT.CMD:
                   ;16531 ;
                   ;16532 ;          SET.FIFO.ADR,             ;START OF THE RL02 READ DATA COMMAND.
                   ;16533 ;          NAD/INIT.FIFO6            ;PRESET FIFOA ADRS CNTR TO OVFLW.
                   ;16534 ;
                   ;16535 ;-----------------------------------------------------------------------
                   ;16536 ;
                   ;16537 ;INIT.FIFO6:
                   ;16538 ;
```

```
;16539 ;        UCMD/CHG.FIFO           ;CHANGE TO FIFOB
;16540 ;
;16541 ;-----------------------------------------------------------------
;16542 ;
;16543 ;INIT.FIFO6A:
;16544 ;
;16545 ;        SET.FIFO.ADR            ;PRESET FIFOB ADRS CNTR TO OVFLW.
;16546 ;
;16547 ;-----------------------------------------------------------------
;16548 ;
;16549 ;INIT.FIFO6B:
;16550 ;
;16551 ;        UCMD/SET.FIFOA,         ;RETURN TO FIFOA
;16552 ;        NAD/WAIT.DRDY1
;16553 ;
;16554 ;-----------------------------------------------------------------
;16555 ;=0
;16556 ;
;16557 ;WAIT.DRDY1:
;16558 ;
;16559 ;        BENO/DRIVE.RDY,         ;WAIT FOR DRIVE READY
;16560 ;        NAD/WAIT.DRDY1
;16561 ;
;16562 ;-----------------------------------------------------------------
;16563 ;=1
;16564 ;
;16565 ;CHGCLK1:
;16566 ;
;16567 ;        ENB.SYS.CLK             ;USE RL02 WRITE CLOCK
;16568 ;
;16569 ;-----------------------------------------------------------------
;16570 ;=0*
;16571 ;
;16572 ;WAIT.SEC.PLS1:
;16573 ;
;16574 ;        BEN1/SECTOR.PLS,        ;WAIT FOR LEADING EDGE OF SECTOR PULSE
;16575 ;        NAD/WAIT.SEC.PLS1
;16576 ;
;16577 ;-----------------------------------------------------------------
;16578 ;=1*
;16579 ;
;16580 ;SEC.PLS.FND1:
;16581 ;
;16582 ;        NAD/WAIT.END.SEC.PLS1   ;SECTOR PULSE HAS BEEN FOUND
;16583 ;
;16584 ;-----------------------------------------------------------------
;16585 ;=1*
;16586 ;
;16587 ;WAIT.END.SEC.PLS1:
;16588 ;
;16589 ;        BEN1/SECTOR.PLS,        ;WAIT FOR END OF SECTOR PULSE
;16590 ;        NAD/WAIT.END.SEC.PLS1
;16591 ;
;16592 ;-----------------------------------------------------------------
;16593 ;=1
```

```
;16594 ;
;16595 ;SETUP.RD.HDR1:
;16596 ;
;16597 ;        LOOP.LOCK/ON,              ;SETUP INITIAL CONDITIONS
;16598 ;        ENB.CLR.BC/ON,             ;FOR READING HEADER.
;16599 ;        FUNC/CLR.DSR,
;16600 ;        LOAD.LOOP.CNTR.[0F1]       ;256-15=241=F1
;16601 ;
;16602 ;----------------------------------------------------------------
;16603 ;
;16604 ;CHGCLK1A:
;16605 ;
;16606 ;        ENB.DSK.CLK,               ;THE MICROSEQUENCER WILL STALL
;16607 ;        LOOP.LOCK/ON,              ;UNTIL SYNC SEEN IS SET (SYNC BIT FOUND)
;16608 ;        ENB.CLR.BC/ON,             ;THEN WE USE CLOCK FROM DATA SEPARATOR.
;16609 ;        GEN.SYNC.[80]
;16610 ;
;16611 ;----------------------------------------------------------------
;16612 ;=0
;16613 ;
;16614 ;READ.HDR1:
;16615 ;
;16616 ;        LOOP.LOCK/ON,              ;SHIFT IN 15 MORE HEADER BITS AND
;16617 ;        DATA.FIELD/ON,             ;COMPARE WITH SERIAL DAR.
;16618 ;        ENB.CLR.BC/ON,
;16619 ;        INCR.LPCNTR/ON,
;16620 ;        BEN0/CNTR.OVFLW,
;16621 ;        NAD/READ.HDR1
;16622 ;
;16623 ;----------------------------------------------------------------
;16624 ;=1
;16625 ;
;16626 ;TEST.HDR.MATCH1:
;16627 ;
;16628 ;        LOOP.LOCK/ON,              ;HEADER HAS BEEN READ.
;16629 ;        UCON/TEST.MATCH,           ;TURN ON TEST MATCH, OFF ENB CLR BC.
;16630 ;        DATA.FIELD/ON,
;16631 ;        NAD/BRANCH.HDR.MATCH1
;16632 ;
;16633 ;----------------------------------------------------------------
;16634 ;
;16635 ;BRANCH.HDR.MATCH1:
;16636 ;
;16637 ;        LOOP.LOCK/ON,
;16638 ;        DATA.FIELD/ON,             ;RESULTS OF HDR COMP NOW IN MISMATCH.
;16639 ;        BEN2/MISMATCH.L,
;16640 ;        NAD/HDR.COMP1
;16641 ;
;16642 ;----------------------------------------------------------------
;16643 ;=1**
;16644 ;
;16645 ;HDR.COMP1:
;16646 ;
;16647 ;        LOOP.LOCK/ON,              ;HEADER COMPARE!
;16648 ;        DATA.FIELD/ON,             ;CONTINUE TO READ THE HDR DENSITY WORD.
```

```
;16649 ;        LOAD.LOOP.CNTR.[OF2],   ;256-14=242=F2
;16650 ;        NAD/RD.DENSITY.WD1
;16651 ;
;16652 ;------------------------------------------------------------------
;16653 ;=0
;16654 ;
;16655 ;RD.DENSITY.WD1:
;16656 ;
;16657 ;        INCR.LPCNTR/ON,         ;CONT TO SHIFT IN THE HDR DENSITY WD.
;16658 ;        LOOP.LOCK/ON,
;16659 ;        DATA.FIELD/ON,
;16660 ;        BENO/CNTR.OVFLW,
;16661 ;        NAD/RD.DENSITY.WD1
;16662 ;
;16663 ;------------------------------------------------------------------
;16664 ;=1
;16665 ;
;16666 ;CRC.PREP1:
;16667 ;
;16668 ;        LOAD.LOOP.CNTR.[OF1],   ;PREPARE TO READ THE CRC WORD.
;16669 ;        LOOP.LOCK/ON,           ;256-15=241=F1
;16670 ;        CHECK.FIELD/ON          ;TURN DFLD OFF AND CFLD ON.
;16671 ;
;16672 ;------------------------------------------------------------------
;16673 ;=0
;16674 ;
;16675 ;RD.CRC.WD1:
;16676 ;
;16677 ;        INCR.LPCNTR/ON,         ;NOW WE SHIFT IN THE CRC WORD
;16678 ;        LOOP.LOCK/ON,
;16679 ;        CHECK.FIELD/ON,
;16680 ;        BENO/CNTR.OVFLW,
;16681 ;        NAD/RD.CRC.WD1
;16682 ;
;16683 ;------------------------------------------------------------------
;16684 ;=1
;16685 ;
;16686 ;CHECK.CRC1:
;16687 ;
;16688 ;        LOOP.LOCK/ON,
;16689 ;        UCON/CHECK.ECC          ;DROP CFLD. CLOCK ECC FLOP
;16690 ;
;16691 ;------------------------------------------------------------------
;16692 ;
;16693 ;TEST.FOR.ERR1:
;16694 ;
;16695 ;        LOOP.LOCK/ON,           ;WAS THERE A CRC ERROR?
;16696 ;        BEN1/ERROR.L
;16697 ;
;16698 ;------------------------------------------------------------------
;16699 ;=0*
;16700 ;
;16701 ;CRC.ERR1:
;16702 ;
;16703 ;        LOOP.LOCK/ON,
;16704 ;        ENB.CPU.CLK,            ;CRC ERROR. TERMINATE AND SET OPI.
;16705 ;        UCMD/SET.OPI,
;16706 ;        NAD/XFER.DONE1
;16707 ;
;16708 ;------------------------------------------------------------------
;16709 ;=1*
;16710 ;
;16711 ;NOT.CRC.ERR1:
;16712 ;
;16713 ;        LOOP.LOCK/ON,
;16714 ;        FUNC/CLR.ECC,           ;CLEAR DC007
;16715 ;        BEN2/F2,                ;BRANCH ON FUNCTION BITS INTO WRT CHK,
;16716 ;        BEN1/F1,                ;RD DATA OR WRT DATA FLOWS.
;16717 ;        NAD/BRANCH.WC.CMD1
;16718 ;
;16719 ;------------------------------------------------------------------
;16720 ;=11*
;16721 ;
;16722 ;BRANCH.RD.CMD6:
;16723 ;
;16724 ;        LOOP.LOCK/ON,           ;CONTINUE VIA READ DATA FLOWS.
;16725 ;        BEN1/FIFO.OVFLW.L,
;16726 ;        NAD/CHGCLK6B
;16727 ;
;16728 ;------------------------------------------------------------------
;16729 ;=0*
;16730 ;
;16731 ;CHGCLK6B:
;16732 ;
;16733 ;        LOOP.LOCK/ON,           ;EVERYTHING IS GO FOR READING DATA FROM
;16734 ;        ENB.SYS.CLK,            ;THIS SECTOR. RUN OFF SYSTEM CLOCK THRU
;16735 ;        LOAD.LOOP.CNTR.[0F5],   ;HEADER POSTAMBLE AND PAST SPLICE.
;16736 ;        NAD/HDR.POST6
;16737 ;
;16738 ;------------------------------------------------------------------
;16739 ;=0
;16740 ;
;16741 ;HDR.POST6:
;16742 ;
;16743 ;        INCR.LPCNTR/ON,         ;SHIFT OVER HEADER POSTAMBLE
;16744 ;        BEN0/CNTR.OVFLW,        ;LOOP LOCK OFF
;16745 ;        NAD/HDR.POST6
;16746 ;
;16747 ;------------------------------------------------------------------
;16748 ;=1
;16749 ;
;16750 ;DATA.PRE6:
;16751 ;
;16752 ;        FUNC/CLR.ECC,
;16753 ;        LOOP.LOCK/ON,           ;NOW WE ARE IN THE DATA PREAMBLE
;16754 ;        LOAD.LOOP.CNTR.[0E0]
;16755 ;
;16756 ;------------------------------------------------------------------
;16757 ;=0
;16758 ;
```

```
;16759 ;DATA.PRE6A:
;16760 ;
;16761 ;        LOOP.LOCK/ON,           ;ALLOW PHASE LOCK LOOP 4 BYTES
;16762 ;        INCR.LPCNTR/ON,         ;OF PREAMBLE TO LOCK UP
;16763 ;        BENO/CNTR.OVFLW,
;16764 ;        NAD/DATA.PRE6A
;16765 ;
;16766 ;-----------------------------------------------------------------
;16767 ;=1
;16768 ;
;16769 ;SETUP.RD.DATA6:
;16770 ;
;16771 ;        LOOP.LOCK/ON,           ;SETUP INITIAL CONDITIONS
;16772 ;        CLR.FIFO.ADR,           ;FOR READING DATA.
;16773 ;        LOAD.LOOP.CNTR.[0F9]
;16774 ;
;16775 ;-----------------------------------------------------------------
;16776 ;
;16777 ;CHGCLK6C:
;16778 ;
;16779 ;        LOOP.LOCK/ON,           ;THE MICROSEQUENCER WILL STALL UNTIL
;16780 ;        ENB.DSK.CLK,            ;SYNC SEEN IS SET.
;16781 ;        GEN.SYNC.[80],          ;THEN WE USE THE DATA SEPARATOR CLK.
;16782 ;        NAD/READ.DATA6
;16783 ;
;16784 ;-----------------------------------------------------------------
;16785 ;=10
;16786 ;
;16787 ;READ.DATA6:
;16788 ;
;16789 ;        LOOP.LOCK/ON,           ;SHIFT IN 7 DATA BITS
;16790 ;        DATA.FIELD/ON,
;16791 ;        INCR.LPCNTR/ON,
;16792 ;        BENO/CNTR.OVFLW,
;16793 ;        NAD/READ.DATA6
;16794 ;
;16795 ;-----------------------------------------------------------------
;16796 ;=11
;16797 ;
;16798 ;STORE.BYTE6:
;16799 ;
;16800 ;        LOOP.LOCK/ON,           ;8TH DATA BIT
;16801 ;        DATA.FIELD/ON,
;16802 ;        ENB.DSR/ON,
;16803 ;        WRT.FIFO/ON,            ;LOAD BYTE INTO FIFO
;16804 ;        LOAD.LOOP.CNTR.[0F9],
;16805 ;        INCR.FIFO.CNTR/ON,
;16806 ;        BEN1/FIFO.MAX.L,
;16807 ;        NAD/READ.DATA6
;16808 ;
;16809 ;-----------------------------------------------------------------
;16810 ;=00
;16811 ;
;16812 ;RD.DATA.CRC6:
;16813 ;
```

```
;16814 ;        LOOP.LOCK/ON,                ;START TO READ THE CRC WORD.
;16815 ;        CHECK.FIELD/ON,              ;DROP DFLD, ASSERT CFLD
;16816 ;        LOAD.LOOP/CNTR.[OF1],
;16817 ;        NAD/RD.DATA.CRC6A
;16818 ;
;16819 ;-----------------------------------------------------------------
;16820 ;=0
;16821 ;
;16822 ;RD.DATA.CRC6A:
;16823 ;
;16824 ;        LOOP.LOCK/ON,
;16825 ;        CHECK.FIELD/ON,
;16826 ;        INCR.LPCNTR/ON,              ;SHIFT IN THE CRC WORD
;16827 ;        BENO/CNTR.OVFLW,
;16828 ;        NAD/RD.DATA.CRC6A
;16829 ;
;16830 ;-----------------------------------------------------------------
;16831 ;=1
;16832 ;
;16833 ;CHECK.CRC6A:
;16834 ;
;16835 ;        LOOP.LOCK/ON,                ;DROP CFLD. CLOCK ECC FLOP
;16836 ;        UCON/CHECK.ECC
;16837 ;
;16838 ;-----------------------------------------------------------------
;16839 ;
;16840 ;TEST.FOR.ERR6A:
;16841 ;
;16842 ;        LOOP.LOCK/ON,                ;SWITCH OVER TO CPU CLOCK
;16843 ;        ENB.CPU.CLK,                 ;FOR FAST INTR OR TERMINATE.
;16844 ;        BEN1/ERROR.L                 ;WAS THERE A CRC ERROR?
;16845 ;
;16846 ;-----------------------------------------------------------------
;16847 ;=0*
;16848 ;
;16849 ;CRC.ERROR6A:
;16850 ;
;16851 ;        NAD/XFER.DONE6               ;CRC ERROR. TERMINATE AND SET CRDY.
;16852 ;
;16853 ;-----------------------------------------------------------------
;16854 ;=1*
;16855 ;
;16856 ;NOT.CRC.ERR6A:
;16857 ;
;16858 ;        CLR.FIFO.ADR,                ;CLEAR THE FIFO ADDRESS COUNTER
;16859 ;        UCMD/SET.XFER.RQST,
;16860 ;        NAD/WAIT.XFER.GRANT6
;16861 ;
;16862 ;-----------------------------------------------------------------
;16863 ;=1*
;16864 ;
;16865 ;WAIT.XFER.GRANT6:
;16866 ;
;16867 ;        BEN1/XFER.REQ,               ;CPU TURNS ON CRDY IF LAST SECTOR
;16868 ;        NAD/WAIT.XFER.GRANT6         ;   THEN SENDS GRANT
```

```
;
;--------------------------------------------------------------------
;=0*
;
;TEST.XFER.DONE6:
;
;       UCMD/CHG.FIFO,          ;SWITCH FIFOS, IN CASE THERE'S MORE DATA
;       BENO/CRDY.L,
;       NAD/XFER.DONE6
;
;--------------------------------------------------------------------
;=0
;
;XFER.DONE6:
;
;       UCMD/SET.CRDY,
;       INCR.DAR/ON,
;       FUNC/SET.INT,           ;CPU SIGNALS END OF TRANSFER.
;       NAD/SET.INT.AND.IDLE
;
;--------------------------------------------------------------------
;=111111110
;
;SET.INT.AND.IDLE:
;
;       FUNC/SET.INT,           ; The purpose of this state is to give CRDY one
;       NAD/IDLE                ; state to reset before asserting interrupt.
;                               ; CRDY is now used in the INT equation.
;
;--------------------------------------------------------------------
;=1
;
;RTN.TO.IDLE:
;IDLE:
;
;       BENO/CRDY.L,            ;LOOK FOR GO (CRDY L)
;       NAD/SETUP.DRV.CHK
;
;--------------------------------------------------------------------
;=1
;
;TEST.OR.RUN:
;
;       BEN2/MAINT,             ;IS THAT YOU, VICKIE?
;       NAD/DRIVE.TYPE
;
;--------------------------------------------------------------------
;=0**
;
;DRIVE.TYPE:
;
;       BENO/R80,               ;WHICH TYPE OF DRIVE WAS SELECTED?
;       NAD/RL02.IFORK
;
;--------------------------------------------------------------------
```

```
                    ;16924 ;=0
                    ;16925 ;
                    ;16926 ;RL02.IFORK:
                    ;16927 ;
                    ;16928 ;        UCMD/SET.FIFOA,
                    ;16929 ;        FUNC/CLR.ECC.ERROR,      ; Enable clear of ECC/CRC errors for new function
                    ;16930 ;        BEN0/F0,                 ;8 WAY INSTRUCTION FORK.
                    ;16931 ;        BEN1/F1,                 ;BRANCH ON 3 FUNCTION BITS IN THE CSR.
                    ;16932 ;        BEN2/F2,
                    ;16933 ;        NAD/NOP.CMD
                    ;16934 ;
                    ;16935 ;----------------------------------------------------------------------
                    ;16936 ;=00C
                    ;16937 ; This was formerly patched to enter diskless loop
                    ;16938 ; This version makes it a true NOP
                    ;16939 ;
                    ;16940 ;NOP.CMD:
                    ;16941 ;
                    ;16942 ;        UCMD/SET.CRDY,           ;SIGNAL END OF FUNCTION.
                    ;16943 ;        FUNC/SET.INT,
                    ;16944 ;        NAD/SET.INT.AND.IDLE
                    ;16945 ;
                    ;16946 ;----------------------------------------------------------------------
                    ;16947
                    ;16948
                    ;16949 WAIT.IDC.TF.7:
U 0D1F, DB,001A     ;16950          SKIP.IF[NO.FAST.INTERRUPT]                 ;wait for xfer req from IDC
U 0D20, 08,D224     ;16951          JMP [IDC.DONE.TF.7]                        ;xfer req set
U 0D21, 88,D1F4     ;16952          JMP [WAIT.IDC.TF.7]                        ;wait some more
                    ;16953
                    ;16954 IDC.DONE.TF.7:
U 0D22, 36,4E15     ;16955          MOV LS[SETCSR.CRDY] TO WR[0]               ;clear CSR F<2:0>
U 0D23, 17,A015     ;16956          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]       ;write the CSR
U 0D24, 90,FA15     ;16957          MISC2 [TRANSFER.GRANT]                     ;send grant to clr the xfer req
                    ;16958
                    ;16959 ;Read data words from FIFO A.
                    ;16960
U 0D25, 17,D815     ;16961          PORT.CONTROL [SEL.FIFO.A]                  ;reading FIFO A
U 0D26, 17,C015     ;16962          PORT.CONTROL [CLEAR.FIFO.CNTR]             ;ensure FIFOA adr.= 0
U 0D27, 90,9015     ;16963          MISC [SET.PORT.I.O]                        ;select IDC
                    ;16964
U 0D28, 97,8C15     ;16965          PORT.READ PORT.ADRS[DATA.WORD]             ;the 1st 4-bytes = don't care
U 0D29, 97,8C15     ;16966          PORT.READ PORT.ADRS[DATA.WORD]             ;load this longword
U 0D2A, 3B,3215     ;16967          MOV PORT TO LS[PORT0]                      ; in a
U 0D2B, 37,3215     ;16968          MOV LS[PORT0] TO WR[0]                     ; working reg.
                    ;16969          TST WR[0],                                 ;is content zero
U 0D2C, 20,0035     ;16970           DT(LONG)&SET.ALU.CC
U 0D2D, 88,D339     ;16971          JMP.IF[EQL] TO [NX.SECTOR]                 ;yes, look for another block
U 0D2E, 97,8C15     ;16972          PORT.READ PORT.ADRS[DATA.WORD]             ;no, load this longword
U 0D2F, 3B,3215     ;16973          MOV PORT TO LS[PORT0]                      ; in a
U 0D30, 37,3215     ;16974          MOV LS[PORT0] TO WR[0]                     ; working reg.
                    ;16975          TST WR[0],                                 ;is content zero
U 0D31, 20,0035     ;16976           DT(LONG)&SET.ALU.CC
U 0D32, 88,D5B1     ;16977          JMP.IF[NEQ] TO [READ.LABEL]                ;no, valid home block found
                    ;16978
```

```
                  ;16979 ;Homeblock is not found, this routine keeps reading all next 2nd sectors on
                  ;16980 ; the first 2 cylinders. Error will be reported if no valid homeblock is found.
                  ;16981
                  ;16982 NX.SECTOR:                                          ;yes, look for another block
U 0D33, 37,9815   ;16983         MOV LS[#26] TO WR[0]                        ;last sector to check for HBLK
U 0D34, 36,0495   ;16984         MOV LS[SAV.WR1] TO WR[1]                    ;no. of sectors counted so far
                  ;16985         CMP WR[0] WITH WR[1],                       ;reaching last sector yet ?
U 0D35, 26,40B5   ;16986          DT(LONG)&SET.ALU.CC
U 0D36, 88,D3A9   ;16987         JMP.IF[EQL] TO [NX.CYL]                     ;yes,seek to the next cylinder
U 0D37, 20,4295   ;16988         INC WR[1]                                   ;no, inc. to the
U 0D38, 20,4295   ;16989         INC WR[1]                                   ; next 2nd sector
U 0D39, 88,D1A4   ;16990         JMP [FIND.HBLK]                             ; and go read and verify it
                  ;16991
                  ;16992 ;Get to the 2nd cylinder when there is no valid homeblock on the first one.
                  ;16993
                  ;16994 NX.CYL:
U 0D3A, B6,4015   ;16995         MOV LS[#1] TO WR[0]                         ;last cyl. to check for HBLK
U 0D3B, 37,8895   ;16996         MOV LS[CYL.CNTR] TO WR[1]                   ;no. of cyl. counted so far
                  ;16997         CMP WR[0] WITH WR[1],                       ;reaching last cyl. yet ?
U 0D3C, 26,40B5   ;16998          DT(LONG)&SET.ALU.CC
U 0D3D, 88,D4D9   ;16999         JMP.IF[EQL] TO [BAD.BLK.ERR]                ;yes, report no valid HBLK
                  ;17000
U 0D3E, 20,4295   ;17001         INC WR[1]                                   ;inc. the cylinder
U 0D3F, BF,8895   ;17002         MOV WR[1] TO LS[CYL.CNTR]                   ; counter by 1
U 0D40, B7,8A95   ;17003         MOV LS[DAR.SEEK] TO WR[1]                   ;DAR fromat it is seeking
U 0D41, C7,4295   ;17004         BIS LS[BIT1] TO WR[1]                       ;set marker bit for DAR word
U 0D42, 40,4E95   ;17005         ADD LS[BIT7] TO WR[1]                       ;inc. current cyl. by 1
U 0D43, 3F,8A95   ;17006         MOV WR[1] TO LS[DAR.SEEK]                   ;store the updated DAR format
U 0D44, 17,A495   ;17007         PORT.WRITE PORT.ADRS[DAR] WITH WR[1]        ;updated DAR seek format
                  ;17008
                  ;17009 ;To bring the heads to the 2nd cylinder from the first one, a seek command has
                  ;17010 ; to be used.
                  ;17011
U 0D45, B7,0495   ;17012         MOV LS[SEEK] TO WR[1]                       ;get the seek command
U 0D46, C7,5095   ;17013         BIS LS[SETCSR.DS1] TO WR[1]                 ;select drive 1
U 0D47, 97,A095   ;17014         PORT.WRITE PORT.ADRS[CSR] WITH WR[1]        ;seek to the next cyl.
                  ;17015
                  ;17016         ;*****************************************************
                  ;17017         ;* Refer to the IDC seek function codes as listed above *
                  ;17018         ;*****************************************************
                  ;17019
                  ;17020 WAIT.IDC.TF.8:
U 0D48, 88,700C   ;17021         JSR [WAIT.CSR.CRDY]                         ;check if CRDY is set
U 0D49, 08,D4B4   ;17022         JMP [IDC.DONE.TF.8]                         ;CRDY set
U 0D4A, 08,D484   ;17023         JMP [WAIT.IDC.TF.8]                         ;wait some more
                  ;17024
                  ;17025 IDC.DONE.TF.8:
U 0D4B, 2F,8295   ;17026         CLR WR[1]                                   ;start from 1st sector
                  ;17027                                                     ; of the 2nd cylinder
U 0D4C, 88,D1A4   ;17028         JMP [FIND.HBLK]                             ;go read the next sector
                  ;17029
                  ;17030 ;Unable to locate a valid homeblock on the first two cylinders.
                  ;17031
                  ;17032 BAD.BLK.ERR:
U 0D4D, B7,9295   ;17033         MOV LS[#5] TO WR[1]
```

```
U 0D4E, 3E,8295         MOV WR[1] TO LS[ERROR.NUMBER]           ;set up error number = 5 in LS
U 0D4F, 2F,8295         CLR WR[1]                               ;error mask value = 0
U 0D50, BE,8A95         MOV WR[1] TO LS[ERROR.MASK]
U 0D51, 36,4C95         MOV LS[RL02] TO WR[1]                   ;store module code in LS to
U 0D52, BE,8C95         MOV WR[1] TO LS[MODULE.NUM]             ; to indicate RL02 module
U 0D53, 36,6F15         MOV LS[NA.EXP.REC] TO WR[2]             ;put "N/A" under data
U 0D54, C7,4915         BIS LS[BIT4] TO WR[2]                   ; and set bit4 to CRD micmon
U 0D55, BE,8115         MOV WR[2] TO LS[CONTROL.STATUS]         ; for printing of error msg.

U 0D56, 10,E015         MISC [SET.CP.ATTN]                      ;report the warning
                                                                ; if micmon is in CRD mode
                WAIT.TF.8:
U 0D57, 88,D574         JMP [WAIT.TF.8]                         ;wait for console to respond

U 0D58, 08,693C         JSR [CHECK.RESULT]                      ;enter here to get a standard
                                                                ; error report for reg. micmon
U 0D59, 08,CEF4         JMP [TEST.TF.4]                         ;loop on err if err bit is set
U 0D5A, 88,D5B4         JMP [READ.LABEL]                        ;no error found, proceed.



                    ;Now it's time to read the home_block sector the label should be in the
                    ; 4th sector of cylinder 0 if there was no prior bad block :

                READ.LABEL:
U 0D5B, 36,0495         MOV LS[SAV.WR1] TO WR[1]                ;valid HBLK. I.D. sector
U 0D5C, 20,4295         INC WR[1]                               ;get to its label sector
U 0D5D, 17,A495         PORT.WRITE PORT.ADRS[DAR] WITH WR[1]    ; with the DAR read format
U 0D5E, B7,0A15         MOV LS[READ.DATA] TO WR[0]              ;set up a read data command
U 0D5F, 47,5015         BIS LS[SETCSR.DS1] TO WR[0]             ;select drive 1
U 0D60, 17,A015         PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;write the CSR

                        ;************************************************************
                        ;* Refer to the IDC read data function codes as listed above *
                        ;************************************************************

                WAIT.IDC.TF.9:
U 0D61, DB,001A         SKIP.IF[NO.FAST.INTERRUPT]              ;wait for xfer req from IDC
U 0D62, 88,D644         JMP [IDC.DONE.TF.9]                     ;xfer req set
U 0D63, 88,D614         JMP [WAIT.IDC.TF.9]                     ;wait some more

                IDC.DONE.TF.9:
U 0D64, 90,FA15         MISC2 [TRANSFER.GRANT]                  ;send grant to clr the xfer req

                   ;Now the home block is in FIFO A, but the 12-byte structure name is located
                   ; at logical bytes " 216 to 227 (dec) " or " D8 to E3 (hex) "

U 0D65, 2F,8015         CLR WR[0]                               ;init. byte counter
U 0D66, 17,D815         PORT.CONTROL [SEL.FIFO.A]               ;reading FIFO A
U 0D67, 17,C015         PORT.CONTROL [CLEAR.FIFO.CNTR]          ;ensure FIFO A adr.=0
U 0D68, 90,9015         MISC [SET.PORT.I.O]                     ;select idc

U 0D69, B7,EB15         MOV LS[#D8] TO WR[2]                    ;FIFO adr. where label
                                                                ; begins

                        ;17089 INC.FIFO.ADR.TF:                                       ;loop for step-up FIFO adr.
U 0D6A, 20,4015         ;17090          INC WR[0]                                     ;step up byte counter
U 0D6B, 17,8815         ;17091          PORT.READ PORT.ADRS[DATA.BYTE]                ; to get to next adr.
                        ;17092          CMP WR[0] WITH WR[2],                         ;reaching the boundary
U 0D6C, 26,4135         ;17093           DT(LONG)&SET.ALU.CC                          ; of the label yet ?
U 0D6D, 08,D6A1         ;17094          JMP.IF[NEQ] TO [INC.FIFO.ADR.TF]              ;no, get to the next adr.
U 0D6E, 08,D6F4         ;17095          JMP [GET.LABEL.TF.1]
                        ;17096
                        ;17097
                        ;17098 ;Get the whole label and put them in 3 long words from LS(8E - 90)
                        ;17099
                        ;17100 GET.LABEL.TF.1:
U 0D6F, 08,716C         ;17101          JSR [LONG.WORD.LABEL]                         ;get 1st long word
U 0D70, 36,0295         ;17102          MOV LS[SAV.WR0] TO WR[1]
U 0D71, 3F,1C95         ;17103          MOV WR[1] TO LS[LABEL.READ1]
                        ;17104
                        ;17105 GET.LABEL.TF.2:
U 0D72, 08,716C         ;17106          JSR [LONG.WORD.LABEL]                         ;get 2nd long word
U 0D73, 36,0295         ;17107          MOV LS[SAV.WR0] TO WR[1]
U 0D74, BF,1E95         ;17108          MOV WR[1] TO LS[LABEL.READ2]
                        ;17109
                        ;17110 GET.LABEL.TF.3:
U 0D75, 08,716C         ;17111          JSR [LONG.WORD.LABEL]                         ;get 3rd long word
U 0D76, 36,0295         ;17112          MOV LS[SAV.WR0] TO WR[1]
U 0D77, 3F,2095         ;17113          MOV WR[1] TO LS[LABEL.READ3]
                        ;17114
                        ;17115
                        ;17116     ;Comparing the label "CRDPACK01   " with fifoa on a long word basis :
                        ;17117
                        ;17118 CMP.4CHAR.TF.1:                                        ;compare the 1st 4_char.
U 0D78, 37,1C15         ;17119          MOV LS[LABEL.READ1] TO WR[0]                  ; from the ones received
U 0D79, 37,2315         ;17120          MOV LS[#43524450] TO WR[2]                    ; with the expected "CRDP"
                        ;17121          CMP WR[0] WITH WR[2],
U 0D7A, 26,4135         ;17122           DT(LONG)&SET.ALU.CC
U 0D7B, 08,DB11         ;17123          JMP.IF[NEQ] TO [WRONG.LABEL]                  ;wrong label detected
                        ;17124
                        ;17125 CMP.4CHAR.TF.2:                                        ;compare the 2nd 4_char.
U 0D7C, B7,1E15         ;17126          MOV LS[LABEL.READ2] TO WR[0]                  ; from the ones received
U 0D7D, B6,2D95         ;17127          MOV LS[#FFFFFF00] TO WR[3]                    ;the last ASCII is not cared
U 0D7E, AF,C615         ;17128          AND WR[3] TO WR[0]                            ; so it's masked out
U 0D7F, 37,2515         ;17129          MOV LS[#41434B20] TO WR[2]                    ; with the expected "ACK "
U 0D80, 2F,C715         ;17130          AND WR[3] TO WR[2]
                        ;17131          CMP WR[0] WITH WR[2],                         ;are they the same ??
U 0D81, 26,4135         ;17132           DT(LONG)&SET.ALU.CC
U 0D82, 08,DB11         ;17133          JMP.IF[NEQ] TO [WRONG.LABEL]                  ;wrong label detected
                        ;17134
U 0D83, 08,DB84         ;17135          JMP [LABEL.FOUND]                             ;label compared
                        ;17136
                        ;17137 WRONG.LABEL1:
U 0D84, 65,A015         ;17138          CLR LS[PREVIOUS.ERROR]                        ;clear previous error number
                        ;17139                                                        ; in LS
U 0D85, 17,CC15         ;17140          PORT.CONTROL [CLEAR.IDC]                      ;init. IDC
U 0D86, 17,CC15         ;17141          PORT.CONTROL [CLEAR.IDC]
U 0D87, 17,CC15         ;17142          PORT.CONTROL [CLEAR.IDC]
                        ;17143
```

```
                  ;17144 TFA.1:
U 0D88, 36,9C15   ;17145          MOV LS[SETCSR.DS0] TO WR[0]               ;drive 0 select bit
U 0D89, 17,A015   ;17146          PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;write CSR to select drive 0
                  ;17147
                  ;17148 WAIT.TFA.1:
U 0D8A, 88,700C   ;17149          JSR [WAIT.CSR.CRDY]                       ;wait for CSR to get ready
U 0D8B, 5B,001E   ;17150          SKIP.IF[SKIP]                             ;ready now
U 0D8C, 88,D8A4   ;17151          JMP [WAIT.TFA.1]                          ;no, wait some more
                  ;17152
U 0D8D, 36,4095   ;17153          MOV LS[#1] TO WR[1]                       ;set WR1 = 1
U 0D8E, 3E,8295   ;17154          MOV WR[1] TO LS[ERROR.NUMBER]             ;set up error number = 1 in LS
U 0D8F, A0,C295   ;17155          COM WR[1]                                 ;create a constant #FFFFFFFE as
U 0D90, BE,8A95   ;17156          MOV WR[1] TO LS[ERROR.MASK]               ; error mask for CSR<0>
U 0D91, 36,4A95   ;17157          MOV LS[IDC] TO WR[1]                      ;store IDC module code in LS
U 0D92, BE,8C95   ;17158          MOV WR[1] TO LS[MODULE.NUM]               ; to indicate module at fault
                  ;17159
U 0D93, 90,9015   ;17160          MISC [SET.PORT.I.O]                       ;select the port to bus
U 0D94, 97,8015   ;17161          PORT.READ PORT.ADRS[CSR]                  ;read CSR back
U 0D95, 3B,3215   ;17162          MOV PORT TO LS[PORT0]                     ; from the port and
U 0D96, 37,3215   ;17163          MOV LS[PORT0] TO WR[0]                    ;  load it in WR0
U 0D97, 36,4095   ;17164          MOV LS[#1] TO WR[1]                       ;expected DRDY bit set
U 0D98, 08,693C   ;17165          JSR [CHECK.RESULT]                        ;is CSR DRDY set ??
U 0D99, 08,D884   ;17166          JMP [TFA.1]                               ;loop on error if LOE is set
                  ;17167
                  ;17168 ;Yes, DRDY is set. i.e. there is a drive in drive 0.
                  ;17169
U 0D9A, 37,3215   ;17170          MOV LS[PORT0] TO WR[0]                    ;reload CSR word in WR0
                  ;17171          BIT WR[0] WITH LS[BIT26],                 ;tst R80 bit for R80 presence
U 0D9B, D9,7435   ;17172           DT(LONG)&SET.ALU.CC
U 0D9C, 88,D9E1   ;17173          JMP.IF[BIT.SET] TO [SETT.DR0.R80]         ;yes, do the next test
                  ;17174
U 0D9D, 08,DB14   ;17175          JMP [WRONG.LABEL]
                  ;17176
                  ;17177 SETT.DR0.R80:
U 0D9E, 36,5015   ;17178          MOV LS[BIT8] TO WR[0]
U 0D9F, 3F,0C15   ;17179          MOV WR[0] TO LS[DR0.R80]                  ;set LS[DR0.R80]
U 0DA0, 2F,8015   ;17180          CLR WR[0]
U 0DA1, BF,0E15   ;17181          MOV WR[0] TO LS[DR0.RL02]                 ;clr. LS[DR0.RL02]
                  ;17182
U 0DA2, 37,1C15   ;17183          MOV LS[LABEL.READ1] TO WR[0]              ; from the ones received
U 0DA3, 37,2315   ;17184          MOV LS[#43524450] TO WR[2]                ; with the expected "CRDP"
                  ;17185          CMP WR[0] WITH WR[2],
U 0DA4, 26,4135   ;17186           DT(LONG)&SET.ALU.CC
U 0DA5, DB,0001   ;17187          SKIP.IF[NEQ]                              ;wrong label detected
                  ;17188
U 0DA6, B7,9295   ;17189          MOV LS[#5] TO WR[1]
U 0DA7, 3E,8295   ;17190          MOV WR[1] TO LS[ERROR.NUMBER]             ;set up error number = 5 in LS
U 0DA8, 2F,8295   ;17191          CLR WR[1]                                 ;error mask value = 0
U 0DA9, BE,8A95   ;17192          MOV WR[1] TO LS[ERROR.MASK]
U 0DAA, B6,4E95   ;17193          MOV LS[PRINT.R80] TO WR[1]                          ;store module code in LS to
U 0DAB, BE,8C95   ;17194          MOV WR[1] TO LS[MODULE.NUM]               ; to indicate RL02 module
U 0DAC, C7,4915   ;17195          BIS LS[BIT4] TO WR[2]                     ; and set bit4 to CRD micmon
U 0DAD, BE,8115   ;17196          MOV WR[2] TO LS[CONTROL.STATUS]           ; for printing of error msg.
                  ;17197
U 0DAE, 10,E015   ;17198          MISC [SET.CP.ATTN]                        ;report the warning
```

```
                   ;17199
                   ;17200 WAIT.TFA.10:
U 0DAF, 08,DAF4    ;17201         JMP [WAIT.TFA.10]                        ;wait for console to respond
                   ;17202                                                  ; and halt
U 0DB0, 08,E064    ;17203         JMP [END.SECTION]
                   ;17204
                   ;17205
                   ;17206
                   ;17207 WRONG.LABEL:
U 0DB1, E5,8A15    ;17208         CLR LS[ERROR.MASK]                       ;no error mask for print out
U 0DB2, 37,9415    ;17209         MOV LS[#6] TO WR[0]
U 0DB3, BE,8215    ;17210         MOV WR[0] TO LS[ERROR.NUMBER]            ;update the error number = 6
U 0DB4, B6,4A15    ;17211         MOV LS[WARNING.IDC] TO WR[0]             ;set up the warning bit (5)
U 0DB5, 3E,8015    ;17212         MOV WR[0] TO LS[CONTROL.STATUS]          ; in the CS word
                   ;17213
U 0DB6, 10,E015    ;17214         MISC [SET.CP.ATTN]                       ;report the warning
                   ;17215
                   ;17216 WAIT.TF.10:
U 0DB7, 08,DB74    ;17217         JMP [WAIT.TF.10]                         ;wait for console to respond
                   ;17218                                                  ; and halt
                   ;17219
                   ;17220 LABEL.FOUND:
                   ;17221 END.TF:                                          ;end of test F
                   ;17222
```

```
.PAGE  "Test 10  RL READ HEADER AND SEEK TESTS (run in all modes - 2nd test in CRD mode)"
;++
;
; Test Description:
;
;       This test will check the RL READ HEADER function by first finding
;       out where the heads are located : (write to the CSR with drive 1
;       select <bit 8> and the read header function <bit 3>. Then the header
;       information will be read from FIFO A). Next, a seek function to a
;       predetermined cylinder will be performed : (the cylinder difference
;       which will be loaded to the DAR with the marker, direction and
;       cylinder difference bits; then CSR will be written with drive select
;       1 <bit 8> and the seek function <bits 1 and 2>). Then to verify that
;       both read header and seek functions work, a read header command will
;       be once again issued to the drive to obtain the current cylinder
;       address which will be compared against the expected address. This
;       test will seek to 2 different cylinders on the disk: one for the
;       innermost, and one for the outermost cylinder.
;
;       Note : If there happens to be bad blocks or headers on either one of
;              the cylinders that the heads are seeking, the header won't
;              compare. The test, instead of reporting an error, will keep
;              reading the next cylinder and see if the header matches until
;              16 tries. If header finally matches with the one expected it
;              will go on seeking the zero cylinder, if not error will be
;              reported.
;
; Assumption:
;
;       Both CPU micros and IDC part I has been run successfully and disk
;       drive 1 is up and spinning and ready to accept commands.
;
; Test Steps:
;
;        1. Write CSR with drive 1 select and read header function bits
;        2. Wait for CRDY to set
;        3. Initialize the counter and the value for cylinder to be sought
;            (the first cylinder value to be sought is the last cylinder)
;        4. Use JSR [SHIFT.HDR] to read the header and shift its cylinder
;            address to the DAR format
;        5. Use JSR [SEEK.DAR] to set up DAR seek format with marker bit
;            set and the appropriate state for dir. bit and the cylinder
;            difference
;        6. Use JSR [SEEK.VERIFY] to do a seek to the desired cylinder
;            and verify if this succeeds.
;        7. Check if header matches the expected one, if not
;        8.       Then check if it has done 16 tries yet, if not
;        9.            Then inc. cyl. counter, and update seek value to
;                           the next available cylinder; and loop back to
;                           step 5
;       10.            Else report ERROR 1
;       11.       Else
;       12. Initialize the counter and the value for cylinder to be sought
;            (the first cylinder value to be sought is the first cylinder)
;       13. Use JSR [SHIFT.HDR] to read the header and shift its cylinder
```

```
                    ;          address to the DAR format
                    ;     14. Use JSR [SEEK.DAR] to set up DAR seek format with marker bit
                    ;          set and the appropriate state for dir. bit and the cylinder
                    ;          difference
                    ;     15. Use JSR [SEEK.VERIFY] to do a seek to the desired cylinder
                    ;          and verify if this succeeds.
                    ;     16. Check if header matches the expected one, if not
                    ;     17.      Then check if it has done 16 tries yet, if not
                    ;     18.           Then inc. cyl. counter, and update seek value to
                    ;                        the next available cylinder; and loop back to
                    ;                        step 14
                    ;     19.      Else report ERROR 2
                    ;     20. Else test ends
                    ;
                    ;
                    ;
                    ; ERRORS:
                    ;
                    ;         Printout:
                    ;              EXPECTED       RECEIVED        MODULE
                    ;              Cyl. adr.      cyl. adr.       RL02
                    ;
                    ;  ERROR 1 - The R/W heads did not seek to the innermost cylinder
                    ;
                    ;  ERROR 2 - The R/W heads did not seek to the outermost cylinder
                    ;
                    ; Debug:
                    ;
                    ; ERROR 1 & 2: Check RL DRIVE COMMAND and the driver where this signal
                    ;              comes from. Also check the DAR serializer.
                    ;--

                    T.10:
U 0DB8, B6,5E15            MOV LS[BEGIN.TEST] TO WR[0]             ;set bit 15 in WR[0] for
                                                                   ; control and status word
U 0DB9, 3E,8015            MOV WR[0] TO LS[CONTROL.STATUS]         ;set bit 15 in control and
                                                                   ; status word - bit 15
                                                                   ; indicates start of test to
                                                                   ; console
U 0DBA, 10,E015            MISC [SET.CP.ATTN]                      ;issue cpu attn to console
                    WAIT.T10.0:
U 0DBB, 08,DBB4            JMP [WAIT.T10.0]                        ;loop here waiting for
                                                                   ; console to respond

U 0DBC, 65,A015            CLR LS[PREVIOUS.ERROR]                  ;clear previous error number
                                                                   ; in LS
U 0DBD, 17,CC15            PORT.CONTROL [CLEAR.IDC]                ;init. IDC
U 0DBE, 17,CC15            PORT.CONTROL [CLEAR.IDC]
U 0DBF, 17,CC15            PORT.CONTROL [CLEAR.IDC]

                    TEST.T10.1:
U 0DC0, B7,0615            MOV LS[READ.HEADER] TO WR[0]            ;get the read_header command
U 0DC1, 47,5015            BIS LS[SETCSR.DS1] TO WR[0]             ;select drive 1
U 0DC2, 17,A015            PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;write the CSR
                                                                   ;MAINT=0,CRDY=0,F<2:0>=100
```

```

                                ;*****************************************************************
                                ;* Refer to the RL02 read header IDC function codes as listed above *
                                ;*                        in test 1                                  *
                                ;*****************************************************************

                 WAIT.IDC.T10.1:
U 0DC3, 88,700C                 JSR [WAIT.CSR.CRDY]                        ;check if CRDY is set
U 0DC4, 08,DC64                 JMP [INIT.T10.2]                           ;yes, CRDY is set now
U 0DC5, 08,DC34                 JMP [WAIT.IDC.T10.1]                       ;no, wait some more


                 ;Read the header word from FIFO A.
                 ;Position the heads to the innermost cylinder of the disk.

                 INIT.T10.2:
U 0DC6, 88,72EC                 JSR [SHIFT.HDR]                            ;shift the header to DAR format
U 0DC7, B7,EC95                 MOV LS[#0000FF80] TO WR[1]                 ;init. value for seek cylinder
U 0DC8, 2F,8515                 CLR WR[2]                                  ;init. cylinder counter

                 TEST.T10.2:
U 0DC9, BF,8495                 MOV WR[1] TO LS[SEEK.CYL]                  ;fetch cylinder to be sought
U 0DCA, 08,745C                 JSR [SEEK.DAR]                             ;setup the dar seek word
U 0DCB, 08,74FC                 JSR [SEEK.VERIFY]                          ;go seek and read header
U 0DCC, B6,0215                 MOV LS[SAV.WR0] TO WR[0]                   ;current cyl. where heads are
U 0DCD, 37,8495                 MOV LS[SEEK.CYL] TO WR[1]                  ;fetch the expected cylinder
                                CMP WR[0] WITH WR[1],                      ;are they equal ?
U 0DCE, 26,40B5                  DT(LONG)&SET.ALU.CC
U 0DCF, 08,DE09                 JMP.IF[EQL] TO [INIT.T10.3]                ;yes, do the next test
U 0DD0, B7,AE95                 MOV LS[#F] TO WR[1]                        ;16 tries to allow bad blocks
                                CMP WR[1] WITH WR[2],                      ;enough cyl. done ?
U 0DD1, A6,4335                  DT(LONG)&SET.ALU.CC
U 0DD2, 88,DD79                 JMP.IF[EQL] TO [T10.ERR.2]                 ;yes, report the error
U 0DD3, 37,8495                 MOV LS[SEEK.CYL] TO WR[1]                  ;reload current seeking cyl.
U 0DD4, C1,4E95                 SUB LS[BIT7] FROM WR[1]                    ;try the next lower cyl.
U 0DD5, 20,4515                 INC WR[2]                                  ;inc. retries counter
U 0DD6, 08,DC94                 JMP [TEST.T10.2]                           ;try another cylinder

                 T10.ERR.2:
U 0DD7, 36,4095                 MOV LS[#1] TO WR[1]
U 0DD8, 3E,8295                 MOV WR[1] TO LS[ERROR.NUMBER]              ;error = 1
U 0DD9, 36,4C95                 MOV LS[RL02] TO WR[1]                      ;RL02 as the module
U 0DDA, BE,8C95                 MOV WR[1] TO LS[MODULE.NUM]                ; at fault
U 0DDB, 37,8495                 MOV LS[SEEK.CYL] TO WR[1]                  ;expected cyl. adr.
U 0DDC, B6,0215                 MOV LS[SAV.WR0] TO WR[0]                   ;received cyl. adr.
U 0DDD, 08,693C                 JSR [CHECK.RESULT]                         ;verify the received cyl. adr.
U 0DDE, 08,DC64                 JMP [INIT.T10.2]                           ;loop on err if err bit set
U 0DDF, 88,DE04                 JMP [INIT.T10.3]                           ;do the next test


                 ;Position the heads to the outermost cylinder of the disk.

                 INIT.T10.3:
U 0DE0, 2F,8295                 CLR WR[1]                                  ;init. value for seek cylinder
U 0DE1, 2F,8515                 CLR WR[2]                                  ;init. cylinder counter
```

```
                    ;17388
                    ;17389 TEST.T10.3:
U ODE2, BF,8495     ;17390          MOV WR[1] TO LS[SEEK.CYL]                       ;fetch cylinder to be sought
U ODE3, 08,745C     ;17391          JSR [SEEK.DAR]                                  ;setup the DAR seek word
U ODE4, 08,74FC     ;17392          JSR [SEEK.VERIFY]                               ;go seek and read header
U ODE5, B6,0215     ;17393          MOV LS[SAV.WRO] TO WR[0]                        ;current cyl. where heads are
U ODE6, 37,8495     ;17394          MOV LS[SEEK.CYL] TO WR[1]                       ;fetch the expected cylinder
                    ;17395          CMP WR[0] WITH WR[1],                           ;are they equal ?
U ODE7, 26,4085     ;17396           DT(LONG)&SET.ALU.CC
U ODE8, 08,DF79     ;17397          JMP.IF[EQL] TO [END.T10]                        ;yes, end of test
U ODE9, B7,AE95     ;17398          MOV LS[#F] TO WR[1]                             ;16 tries to allow bad blocks
                    ;17399          CMP WR[1] WITH WR[2],                           ;enough cyl. done ?
U ODEA, A6,4335     ;17400           DT(LONG)&SET.ALU.CC
U ODEB, 88,DF09     ;17401          JMP.IF[EQL] TO [T10.ERR.3]                      ;yes, report the error
U ODEC, 37,8495     ;17402          MOV LS[SEEK.CYL] TO WR[1]                       ;reload current seeking cyl.
U ODED, 40,4E95     ;17403          ADD LS[BIT7] TO WR[1]                           ;try the next higher cyl.
U ODEE, 20,4515     ;17404          INC WR[2]                                       ;inc. retry counter
U ODEF, 08,DE24     ;17405          JMP [TEST.T10.3]                                ;try another cylinder
                    ;17406
                    ;17407 T10.ERR.3:
U ODF0, B6,4295     ;17408          MOV LS[#2] TO WR[1]
U ODF1, 3E,8295     ;17409          MOV WR[1] TO LS[ERROR.NUMBER]                   ;error = 2
U ODF2, 37,8495     ;17410          MOV LS[SEEK.CYL] TO WR[1]                       ;expected cyl. adr.
U ODF3, B6,0215     ;17411          MOV LS[SAV.WRO] TO WR[0]                        ;received cyl. adr.
U ODF4, 08,693C     ;17412          JSR [CHECK.RESULT]                              ;verify the received cyl. adr.
U ODF5, 08,DE24     ;17413          JMP [TEST.T10.3]                                ;loop on err if err bit set
U ODF6, 88,DF74     ;17414          JMP [END.T10]                                   ;end test 10
                    ;17415
                    ;17416 END.T10:                                                ;end of test 10
U ODF7, 08,E064     ;17417          JMP [END.SECTION]                               ;end of CRD optional tests
                    ;17418
```

```
                   .PAGE  "Test 11  IDC SYNCHRONIZATION WITH RL02 PROGRAM (Manual mode only)"
                   ;++
                   ;
                   ; Program Description:
                   ;
                   ;        This program is not a test but rather a software setup so that
                   ;        Manufacturing can eliminate cumbersome setups before it can
                   ;        bring IDC in synchronization with a RL02 disk drive. This
                   ;        program will select drive 1 which presumably has a RL02 drive
                   ;        up and spinning; it will then call in a function of the RL READ
                   ;        HEADER and keep looping infinitely in its IDC function codes.
                   ;
                   ;   *** In order to avoid Micmon timeout, type SET LOOP upon mic prompt
                   ;        before entering into the program.
                   ;
                   ; Program Steps:
                   ;
                   ;        1. Select drive 1 and Invoke RL02 Read Header function by
                   ;             setting CSR <8 & 3>
                   ;        2. Wait for CRDY to set
                   ;        3. Go back to step 1
                   ;
                   ;--

                   T.11:
U 0DF8, B6,5E15            MOV LS[BEGIN.TEST] TO WR[0]               ;set bit 15 in WR[0] for
                                                                     ; control and status word
U 0DF9, 3E,8015            MOV WR[0] TO LS[CONTROL.STATUS]           ;set bit 15 in control and
                                                                     ; status word - bit 15
                                                                     ; indicates start of test to
                                                                     ; console
U 0DFA, 10,E015            MISC [SET.CP.ATTN]                        ;issue CPU ATTN to console

                   WAIT.T11.0:
U 0DFB, 88,DFB4            JMP [WAIT.T11.0]                          ;loop here waiting for
                                                                     ; console to respond

U 0DFC, 65,A015            CLR LS[PREVIOUS.ERROR]                    ;clear previous error number
                                                                     ; in LS

U 0DFD, 17,CC15            PORT.CONTROL [CLEAR.IDC]                  ;init. IDC
U 0DFE, 17,CC15            PORT.CONTROL [CLEAR.IDC]
U 0DFF, 17,CC15            PORT.CONTROL [CLEAR.IDC]

                   ;Initiate the RL02 read header function and make it loop around that infinitely :

                   T11.1:
U 0E00, B7,0615            MOV LS[READ.HEADER] TO WR[0]              ;select read header
U 0E01, 47,5015            BIS LS[SETCSR.DS1] TO WR[0]               ;select the RL02 at drive 0
U 0E02, 17,A015            PORT.WRITE PORT.ADRS[CSR] WITH WR[0]      ;start read header function

                           ;*****************************************************************
                           ;* REFER TO RL02 READ HEADER IDC FUNCTION CODES AS LISTED IN TEST 5 *
                           ;*****************************************************************

                   ;17474 WAIT.IDC.T11.1:
U 0E03, 88,700C    ;17475         JSR [WAIT.CSR.CRDY]                     ;wait for CRDY to set
U 0E04, 08,E004    ;17476         JMP [T11.1]                             ;read header again
U 0E05, 08,E034    ;17477         JMP [WAIT.IDC.T11.1]                    ;wait some more
                   ;17478
                   ;17479    ; *** This program should loop infinitely : ***
```

```
                  ;17480 .PAGE
                  ;17481
                  ;17482 END.SECTION:
U 0E06, 36,5C15   ;17483         MOV LS[BIT14] TO WR[0]                    ;SET UP WR[0] FOR END OF SECTION
U 0E07, 3E,8015   ;17484         MOV WR[0] TO LS[CONTROL.STATUS]           ;SET UP CONTROL AND STATUS
                  ;17485                                                   ; WORD TO REPORT END OF SECTION
U 0E08, 10,E015   ;17486         MISC [SET.CP.ATTN]                        ;ISSUE CPU ATTN TO CONSOLE
                  ;17487 WAIT.EOS:
U 0E09, 08,E094   ;17488         JMP [WAIT.EOS]                            ;LOOP HERE WAITING FOR CONSOLE
                  ;17489                                                   ; TO RESPOND
                  ;17490
                  ;17491
                  ;17492
                  ;17493
```

```
;                              Location / Line Number Index

;Location       0/8     1/9     2/A     3/B     4/C     5/D     6/E     7/F

C 0000          2449:   2450:   2451:   2452:   2453:   2454:   2455:   2456:
C 0008          2457:   2458:   2459:   2460:   2461:   2462:   2463:   2464:
C 0010          2468:   2469:   2470:   2471:   2472:   2473:   2474:   2475:
C 0018          2476:   2477:   2478:   2479:   2480:   2481:   2482:   2483:
C 0020          2492:   2493:   2494:   2495:   2496:   2497:   2498:   2499:
C 0028          2500:   2501:   2502:   2503:   2504:   2505:   2506:   2507:
C 0030          2508:   2509:   2510:   2511:   2512:   2513:   2514:   2515:
C 0038          2516:   2517:   2518:   2519:   2520:   2521:   2522:   2523:
C 0040          2532:   2533:   2534:   2535:   2536:   2537:   2538:   2539:
C 0048          2540:   2541:   2542:   2543:   2544:   2545:   2546:   2547:
C 0050          2548:   2549:   2550:   2551:   2552:   2553:   2554:   2555:
C 0058          2556:   2557:   2558:   2559:   2560:   2561:   2562:   2563:
C 0060          2571:   2572:   2573:   2574:   2575:   2576:   2577:   2578:
C 0068          2579:   2580:   2581:   2582:   2583:   2584:   2585:   2586:
C 0070          2587:   2588:   2589:   2590:   2591:   2592:   2593:   2594:
C 0078          2595:   2596:   2597:   2598:   2599:   2600:   2601:   2602:
C 0080          2609:   2611:   2613:   2615:   2618:   2620:   2622:   2624:
C 0088          2627:   2629:   2631:   2633:   2636:   2638:   2640:   2642:
C 0090          2645:   2647:   2649:   2651:   2654:   2656:   2658:   2660:
C 0098          2663:   2665:   2667:   2669:   2672:   2674:   2676:   2678:
C 00A0          2681:   2683:   2685:   2687:   2690:   2692:   2694:   2696:
C 00A8          2699:   2701:   2703:   2705:   2708:   2710:   2712:   2714:
C 00B0          2717:   2719:   2721:   2723:   2726:   2728:   2730:   2732:
C 00B8          2735:   2737:   2739:   2741:   2744:   2746:   2748:   2750:
C 00C0          2758:   2759:   2760:   2761:   2762:   2763:   2764:   2765:
C 00C8          2768:   2769:   2770:   2771:   2772:   2773:   2774:   2775:
C 00D0          2778:   2779:   2780:   2781:   2782:   2783:   2784:   2785:
C 00D8          2788:   2789:   2790:   2791:   2792:   2793:   2794:   2795:
C 00E0          2799:   2800:   2801:   2802:   2803:   2804:   2805:   2806:
C 00E8          2808:   2809:   2810:   2811:   2812:   2813:   2814:   2815:
C 00F0          2819:   2820:   2821:   2822:   2823:   2824:   2825:   2826:
C 00F8          2829:   2830:   2831:   2832:   2833:   2834:   2835:   2836:
C 0100          2844:   2845:   2846:   2847:   2849:   2850:   2851:   2852:
C 0108          2854:   2855:   2856:   2857:   2859:   2860:   2861:   2862:
C 0110          2865:   2866:   2867:   2868:   2870:   2871:   2872:   2873:
C 0118          2875:   2876:   2877:   2878:   2880:   2881:   2882:   2883:
C 0120          2886:   2887:   2888:   2889:   2891:   2892:   2893:   2894:
C 0128          2896:   2897:   2898:   2899:   2901:   2902:   2903:   2904:
C 0130          2907:   2908:   2909:   2910:   2912:   2913:   2914:   2915:
C 0138          2917:   2918:   2919:   2920:   2922:   2923:   2924:   2925:
C 0140          2928:   2929:   2930:   2931:   2933:   2934:   2935:   2936:
C 0148          2938:   2939:   2940:   2941:   2943:   2944:   2945:   2946:
C 0150          2949:   2950:   2951:   2952:   2954:   2955:   2956:   2957:
C 0158          2959:   2960:   2961:   2962:   2964:   2965:   2966:   2967:
C 0160          2970:   2971:   2972:   2973:   2975:   2976:   2977:   2978:
C 0168          2980:   2981:   2982:   2983:   2985:   2986:   2987:   2988:
C 0170          2991:   2992:   2993:   2994:   2996:   2997:   2998:   2999:
C 0178          3001:   3002:   3003:   3004:   3006:   3007:   3008:   3009:
C 0180          3019:   3020:   3021:   3022:   3024:   3025:   3026:   3027:
C 0188          3029:   3030:   3031:   3032:   3034:   3035:   3036:   3037:
C 0190          3040:   3041:   3042:   3043:   3045:   3046:   3047:   3048:
C 0198          3050:   3051:   3052:   3053:   3055:   3056:   3057:   3058:
C 01A0          3061:   3062:   3063:   3064:   3066:   3067:   3068:   3069:
```

```
;Location        0/8      1/9      2/A      3/B      4/C      5/D      6/E      7/F

C 01A8           3071:    3072:    3073:    3074:    3076:    3077:    3078:    3079:
C 01B0           3082:    3083:    3084:    3085:    3087:    3088:    3089:    3090:
C 01B8           3092:    3093:    3094:    3095:    3097:    3098:    3099:    3100:
C 01C0           3103:    3104:    3105:    3106:    3108:    3109:    3110:    3111:
C 01C8           3113:    3114:    3115:    3116:    3118:    3119:    3120:    3121:
C 01D0           3124:    3125:    3126:    3127:    3129:    3130:    3131:    3132:
C 01D8           3134:    3135:    3136:    3137:    3139:    3140:    3141:    3142:
C 01E0           3145:    3146:    3147:    3148:    3150:    3151:    3152:    3153:
C 01E8           3155:    3156:    3157:    3158:    3160:    3161:    3162:    3163:
C 01F0           3166:    3167:    3168:    3169:    3171:    3172:    3173:    3174:
C 01F8           3176:    3177:    3178:    3179:    3181:    3182:    3183:    3184:
```

```
;                                 Location / Line Number Index

;Location        0/8     1/9     2/A     3/B     4/C     5/D     6/E     7/F

U 0000           4861:   4698:   4699:   4700:   4701:   4702:   4703:   4704:
U 0008           4705:   4706:   4707:   4708:   4709:   4710:   4711:   4712:
U 0010           4713:   4714:   4715:   4717:   4719:   4721:   4723:   4725:
U 0018           4727:   4729:   4731:   4733:   4735:   4737:   4739:   4741:
U 0020           4743:   4745:   4747:   4749:   4751:   4753:   4755:   4757:
U 0028           4759:   4761:   4763:   4765:   4767:   4769:   4771:   4773:
U 0030           4775:   4777:   4779:   4781:   4783:   4785:   4787:   4789:
U 0038           4791:   4793:   4795:   4797:   4799:   4801:   4803:   4805:
U 0040           4807:   4809:   4811:   4813:   4815:   4817:   4819:   4821:
U 0048           4823:   4825:   4827:   4829:   4831:   4833:   4835:   4837:
U 0050           4870:   4872:   4874:   4877:   4880:   4884:   4886:   4888:
U 0058           4890:   4892:   4894:   4896:   4898:   4900:   4902:   4904:
U 0060           4906:   4908:   4910:   4912:   4914:   4916:   4917:   4918:
U 0068           4919:   4920:   4921:   4922:   4923:   4924:   4925:   4926:
U 0070           4956:   4958:   4960:   4961:   4963:   4964:   4966:   4967:
U 0078           4969:   4970:   4972:   4973:   4975:   4976:   4978:   4979:
U 0080           4981:   4982:   4984:   4985:   4987:   4988:   4990:   4991:
U 0088           4993:   4994:   4996:   4997:   4999:   5000:   5002:   5003:
U 0090           5005:   5006:   5008:   5009:   5011:   5012:   5014:   5015:
U 0098           5017:   5018:   5020:   5021:   5023:   5024:   5026:   5027:
U 00A0           5029:   5030:   5032:   5033:   5035:   5036:   5038:   5039:
U 00A8           5041:   5042:   5044:   5045:   5047:   5048:   5050:   5051:
U 00B0           5053:   5054:   5056:   5057:   5059:   5060:   5062:   5063:
U 00B8           5065:   5066:   5068:   5069:   5071:   5072:   5074:   5075:
U 00C0           5077:   5078:   5080:   5081:   5083:   5084:   5086:   5087:
U 00C8           5089:   5090:   5092:   5093:   5095:   5096:   5098:   5099:
U 00D0           5101:   5102:   5104:   5105:   5107:   5108:   5110:   5111:
U 00D8           5113:   5114:   5116:   5117:   5119:   5120:   5122:   5123:
U 00E0           5125:   5126:   5128:   5129:   5131:   5132:   5134:   5135:
U 00E8           5137:   5138:   5140:   5141:   5143:   5144:   5146:   5147:
U 00F0           5149:   5150:   5152:   5153:   5155:   5156:   5158:   5159:
U 00F8           5161:   5162:   5164:   5165:   5167:   5168:   5170:   5171:
U 0100           5173:   5174:   5176:   5177:   5179:   5180:   5182:   5183:
U 0108           5185:   5186:   5188:   5189:   5191:   5192:   5194:   5195:
U 0110           5197:   5198:   5200:   5201:   5203:   5204:   5206:   5207:
U 0118           5209:   5210:   5212:   5213:   5215:   5216:   5218:   5219:
U 0120           5221:   5222:   5224:   5225:   5227:   5228:   5230:   5231:
U 0128           5233:   5234:   5236:   5237:   5239:   5240:   5242:   5243:
U 0130           5245:   5246:   5248:   5249:   5251:   5252:   5254:   5255:
U 0138           5257:   5258:   5260:   5261:   5263:   5264:   5266:   5267:
U 0140           5269:   5270:   5272:   5273:   5275:   5276:   5278:   5279:
U 0148           5281:   5282:   5284:   5285:   5287:   5288:   5290:   5291:
U 0150           5293:   5294:   5296:   5297:   5299:   5300:   5302:   5303:
U 0158           5305:   5306:   5308:   5309:   5311:   5312:   5314:   5315:
U 0160           5317:   5318:   5327:   5329:   5331:   5332:   5334:   5335:
U 0168           5337:   5338:   5340:   5341:   5343:   5344:   5346:   5347:
U 0170           5349:   5350:   5352:   5353:   5355:   5356:   5358:   5359:
U 0178           5361:   5362:   5364:   5365:   5367:   5368:   5370:   5371:
U 0180           5373:   5374:   5376:   5377:   5379:   5380:   5382:   5383:
U 0188           5385:   5386:   5388:   5389:   5391:   5392:   5394:   5395:
U 0190           5397:   5398:   5400:   5401:   5403:   5404:   5406:   5407:
U 0198           5409:   5410:   5412:   5413:   5415:   5416:   5418:   5419:
U 01A0           5421:   5422:   5424:   5425:   5427:   5428:   5430:   5431:
```

```
;                                   Location / Line Number Index

;Location        0/8     1/9     2/A     3/B     4/C     5/D     6/E     7/F

U 01A8           5433:   5434:   5436:   5437:   5439:   5440:   5442:   5443:
U 01B0           5445:   5446:   5448:   5449:   5451:   5452:   5454:   5455:
U 01B8           5457:   5458:   5460:   5461:   5463:   5464:   5466:   5467:
U 01C0           5469:   5470:   5472:   5473:   5475:   5476:   5478:   5479:
U 01C8           5481:   5482:   5484:   5485:   5487:   5488:   5490:   5491:
U 01D0           5493:   5494:   5496:   5497:   5499:   5500:   5502:   5503:
U 01D8           5505:   5506:   5508:   5509:   5511:   5512:   5514:   5515:
U 01E0           5517:   5518:   5520:   5521:   5523:   5524:   5526:   5527:
U 01E8           5529:   5530:   5532:   5533:   5535:   5536:   5538:   5539:
U 01F0           5541:   5542:   5544:   5545:   5547:   5548:   5550:   5551:
U 01F8           5553:   5554:   5556:   5557:   5559:   5560:   5562:   5563:
U 0200           5565:   5566:   5568:   5569:   5571:   5572:   5574:   5575:
U 0208           5577:   5578:   5580:   5581:   5583:   5584:   5586:   5587:
U 0210           5589:   5590:   5592:   5593:   5595:   5596:   5598:   5599:
U 0218           5601:   5602:   5604:   5605:   5607:   5608:   5610:   5611:
U 0220           5613:   5614:   5616:   5617:   5619:   5620:   5622:   5623:
U 0228           5625:   5626:   5628:   5629:   5631:   5632:   5634:   5635:
U 0230           5637:   5638:   5640:   5641:   5643:   5644:   5646:   5647:
U 0238           5649:   5650:   5652:   5653:   5655:   5656:   5658:   5659:
U 0240           5661:   5662:   5664:   5665:   5667:   5668:   5670:   5671:
U 0248           5673:   5674:   5676:   5677:   5679:   5680:   5682:   5683:
U 0250           5685:   5686:
U 0258 - 05FF Unused
U 0600           5709:   5710:   5712:   5715:   5716:   5717:   5731:   5732:
U 0608           5793:   5795:   5797:   5799:   5800:   5855:   5856:   5858:
U 0610           5860:   5862:   5865:   5867:   5868:   5869:   5873:   5874:
U 0618           5875:   5879:   5880:   5881:   5883:   5886:   5887:   5942:
U 0620           5944:   5945:   5948:   5950:   5954:   5956:   5960:   5966:
U 0628           5968:   5969:   5975:   5977:   5979:   5980:   5981:   5987:
U 0630           5988:   5989:   5990:   5991:   5992:   5993:   5996:   5998:
U 0638           5999:   6003:   6053:   6055:   6057:   6059:   6062:   6063:
U 0640           6065:   6066:   6121:   6123:   6124:   6127:   6129:   6133:
U 0648           6135:   6139:   6145:   6147:   6148:   6154:   6155:   6156:
U 0650           6157:   6158:   6159:   6162:   6212:   6214:   6217:   6219:
U 0658           6222:   6223:   6225:   6226:   6274:   6279:   6281:   6283:
U 0660           6284:   6285:   6335:   6337:   6340:   6344:   6346:   6347:
U 0668           6386:   6387:   6388:   6391:   6392:   6393:   6396:   6397:
U 0670           6398:   6401:   6402:   6403:   6406:   6407:   6411:   6412:
U 0678           6413:   6452:   6453:   6454:   6458:   6459:   6460:   6464:
U 0680           6465:   6466:   6470:   6471:   6475:   6476:   6480:   6483:
U 0688           6485:   6532:   6533:   6534:   6535:   6536:   6583:   6584:
U 0690           6585:   6586:   6587:   6655:   6657:   6660:   6661:   6663:
U 0698           6665:   6666:   6667:   6669:   6673:   6675:   6677:   6681:
U 06A0           6682:   6685:   6686:   6688:   6690:   6691:   6693:   6695:
U 06A8           6700:   6701:   6703:   6705:   6706:   6712:   6713:   6716:
U 06B0           6717:   6718:   6723:   6724:   6726:   6727:   6729:   6731:
U 06B8           6732:   6734:   6736:   6738:   6740:   6742:   6749:   6750:
U 06C0           6752:   6755:   6768:   6769:   6771:   6772:   6773:   6774:
U 06C8           6775:   6776:   6777:   6778:   6779:   6783:   6786:   6787:
U 06D0           6788:   6789:   6790:   6791:   6792:   6793:   6794:   6795:
U 06D8           6796:   6797:   6798:   6799:   6800:   6801:   6804:   6807:
U 06E0           6810:   6812:   6813:   6814:   6815:   6818:   6821:   6824:
U 06E8           6826:   6827:   6828:   6830:   6831:   6832:   6834:   6836:
```

```
;Location        0/8     1/9     2/A     3/B     4/C     5/D     6/E     7/F

U 06F0          6837:   6838:   6839:   6842:   6843:   6846:   6848:   6849:
U 06F8 - 06FF Unused
U 0700          6897:   6898:   6899:   6900:   6902:   6903:   6904:   6905:
U 0708          6948:   6949:   6950:   6951:   6953:   6954:   6955:   6957:
U 0710          7002:   7003:   7004:   7005:   7006:   7007:   7051:   7054:
U 0718          7055:   7057:   7058:   7059:   7061:   7064:   7065:   7067:
U 0720          7068:   7069:   7070:   7073:   7076:   7077:   7079:   7080:
U 0728          7081:   7082:   7085:   7086:   7087:   7089:   7133:   7134:
U 0730          7135:   7136:   7137:   7138:   7139:   7140:   7141:   7142:
U 0738          7143:   7144:   7145:   7146:   7147:   7148:   7149:   7150:
U 0740          7151:   7152:   7153:   7154:   7155:   7201:   7202:   7204:
U 0748          7205:   7206:   7208:   7209:   7211:   7212:   7213:   7258:
U 0750          7259:   7261:   7262:   7263:   7271:   7272:   7273:   7277:
U 0758          7282:   7283:   7284:   7293:   7294:   7295:   7298:   7300:
U 0760          7301:   7303:   7359:   7361:   7365:   7367:   7369:   7372:
U 0768          7373:   7374:   7376:   7377:   7379:   7383:   7384:   7385:
U 0770          7386:   7387:   7388:   7390:   7391:   7392:   7393:   7394:
U 0778          7395:   7397:   7398:   7399:   7400:   7401:   7402:   7404:
U 0780          7405:   7406:   7407:   7408:   7409:   7411:   7412:   7413:
U 0788          7414:   7415:   7416:   7418:   7419:   7420:   7421:   7422:
U 0790          7423:   7496:   7498:   7502:   7504:   7507:   7508:   7509:
U 0798          7511:   7515:   7516:   7519:   7520:   7522:   7523:   7524:
U 07A0          7525:   7526:   7528:   7529:   7532:   7533:   7536:   7537:
U 07A8          7539:   7540:   7541:   7542:   7543:   7545:   7546:   7549:
U 07B0          7550:   7551:   7553:   7556:   7558:   7559:   7561:   7562:
U 07B8          7563:   7564:   7565:   7566:   7568:   7572:   7573:   7574:
U 07C0          7575:   7577:   7658:   7660:   7664:   7667:   7669:   7671:
U 07C8          7672:   7673:   7676:   7677:   7680:   7681:   7682:   7684:
U 07D0          7685:   7686:   7687:   7688:   7689:   7691:   7692:   7693:
U 07D8          7694:   7695:   7696:   7697:   7701:   7703:   7704:   7706:
U 07E0          7707:   7708:   7709:   7710:   7713:   7714:   7715:   7716:
U 07E8          7719:   7720:   7721:   7724:   7725:   7728:   7729:   7730:
U 07F0          7732:   7733:   7734:   7735:   7736:   7737:   7738:   7740:
U 07F8          7741:   7742:   7743:   7744:   7745:   7746:   7750:   7751:
U 0800          7752:   7753:   7754:   7755:   7756:   7757:   7758:   7759:
U 0808          7760:   7762:   7763:   7767:   7769:   7770:   7774:   7775:
U 0810          7776:   7777:   7778:   7779:   7784:   7785:   7786:   7787:
U 0818          7788:   7789:   7790:   7791:   7878:   7880:   7884:   7887:
U 0820          7889:   7891:   7892:   7893:   7896:   7897:   7898:   7901:
U 0828          7902:   7904:   7905:   7906:   7907:   8119:   8120:   8121:
U 0830          8122:   8123:   8124:   8125:   8127:   8128:   8133:   8134:
U 0838          8136:   8137:   8138:   8139:   8140:   8141:   8142:   8147:
U 0840          8148:   8149:   8150:   8151:   8156:   8157:   8158:   8159:
U 0848          8160:   8161:   8162:   8163:   8167:   8168:   8169:   8170:
U 0850          8171:   8172:   8173:   8174:   8175:   8176:   8270:   8272:
U 0858          8276:   8279:   8282:   8284:   8285:   8286:   8291:   8292:
U 0860          8293:   8595:   8596:   8597:   8598:   8599:   8600:   8601:
U 0868          8603:   8604:   8607:   8608:   8610:   8611:   8612:   8613:
U 0870          8614:   8615:   8616:   8621:   8622:   8623:   8624:   8625:
U 0878          8630:   8631:   8632:   8633:   8634:   8635:   8636:   8637:
U 0880          8643:   8644:   8645:   8646:   8648:   8650:   8653:   8654:
U 0888          8655:   8866:   8867:   8870:   8871:   8873:   8874:   8875:
U 0890          8876:   8877:   8878:   8879:   8884:   8885:   8886:   8887:
```

```
;Location       0/8     1/9     2/A     3/B     4/C     5/D     6/E     7/F

U 0898          8888:   8894:   8895:   8896:   8903:   8904:   8905:   8908:
U 08A0          8909:   8910:   8911:   8912:   8913:   8914:   8918:   8920:
U 08A8          8921:   8922:   8923:   8924:   8929:   8930:   8931:   8932:
U 08B0          8933:   8934:   8935:   8936:   8937:   8938:   8939:   9038:
U 08B8          9040:   9044:   9047:   9053:   9054:   9055:   9056:   9057:
U 08C0          9058:   9060:   9062:   9063:   9064:   9067:   9069:   9070:
U 08C8          9075:   9076:   9077:   9079:   9080:   9081:   9088:   9089:
U 08D0          9090:   9097:   9098:   9099:   9104:   9105:   9106:   9111:
U 08D8          9115:   9116:   9118:   9119:   9120:   9127:   9128:   9129:
U 08E0          9134:   9136:   9137:   9143:   9144:   9146:   9147:   9148:
U 08E8          9623:   9624:   9625:   9626:   9627:   9628:   9629:   9631:
U 08F0          9632:   9635:   9636:   9638:   9639:   9640:   9641:   9642:
U 08F8          9643:   9644:   9649:   9650:   9651:   9652:   9653:   9656:
U 0900          9661:   9662:   9663:   9664:   9666:   9670:   9671:   9673:
U 0908          9674:   9675:   9681:   9682:   9683:   9686:   9687:   9688:
U 0910          9691:   9692:   9693:   9699:   9700:   9702:   9703:   9706:
U 0918          9707:   9708:   9709:   9710:   9712:   9713:   9714:   9717:
U 0920          9718:   9719:   9720:   9721:   9722:   9724:   9726:   9727:
U 0928          9731:   9732:   9733:   9734:   9735:   9738:   9739:   9741:
U 0930          9744:   9746:   9826:   9828:   9832:   9835:   9838:   9840:
U 0938          9841:   9842:   9848:   9849:   9850:   9855:   9856:   9857:
U 0940          9862:   9863:   9864:   9865:   9866:   9867:   9868:   9869:
U 0948          9870:   9875:   9876:   9877:   9880:   9884:   9885:   9886:
U 0950          9889:   9890:   9891:   9892:   9895:   9896:   9898:   9899:
U 0958          9904:   9905:   9907:   9908:   9909:   10443:  10444:  10445:
U 0960          10446:  10447:  10448:  10449:  10451:  10452:  10457:  10458:
U 0968          10460:  10461:  10462:  10463:  10464:  10465:  10466:  10471:
U 0970          10472:  10473:  10474:  10475:  10478:  10484:  10485:  10486:
U 0978          10487:  10490:  10491:  10493:  10494:  10498:  10499:  10501:
U 0980          10502:  10503:  10510:  10511:  10512:  10517:  10518:  10519:
U 0988          10520:  10521:  10522:  10523:  10524:  10525:  10530:  10531:
U 0990          10532:  10533:  10534:  10535:  10536:  10537:  10538:  10539:
U 0998          10545:  10547:  10548:  10549:  10550:  10551:  10552:  10553:
U 09A0          10554:  10623:  10625:  10629:  10632:  10635:  10637:  10638:
U 09A8          10639:  10645:  10646:  10647:  10652:  10653:  10654:  10659:
U 09B0          10660:  10661:  10662:  10663:  10664:  10665:  10666:  10667:
U 09B8          10672:  10673:  10674:  10679:  10680:  10681:  10682:  10685:
U 09C0          10686:  10688:  10689:  10694:  10695:  10697:  10698:  10699:
U 09C8          10706:  10707:  10708:  10713:  10718:  10719:  10720:  10721:
U 09D0          10724:  10725:  10727:  10728:  10732:  10733:  10735:  10736:
U 09D8          10737:  11254:  11255:  11256:  11257:  11259:  11260:  11261:
U 09E0          11262:  11263:  11264:  11265:  11266:  11268:  11269:  11274:
U 09E8          11275:  11277:  11278:  11279:  11280:  11281:  11282:  11283:
U 09F0          11288:  11289:  11290:  11291:  11292:  11293:  11296:  11297:
U 09F8          11394:  11396:  11400:  11403:  11406:  11408:  11409:  11410:
U 0A00          11413:  11415:  11416:  11419:  11420:  11421:  11581:  11582:
U 0A08          11583:  11584:  11585:  11586:  11587:  11589:  11590:  11593:
U 0A10          11594:  11596:  11597:  11598:  11599:  11600:  11601:  11602:
U 0A18          11607:  11608:  11609:  11610:  11611:  11616:  11617:  11618:
U 0A20          11619:  11620:  11621:  11622:  11623:  11627:  11628:  11629:
U 0A28          11630:  11631:  11632:  11633:  11634:  11635:  11636:  11712:
U 0A30          11714:  11718:  11721:  11724:  11726:  11727:  11728:  11731:
U 0A38          11733:  11734:  11738:  11739:  11744:  11745:  11750:  11751:
```

```
;                                    Location / Line Number Index

;Location      0/8     1/9     2/A     3/B     4/C     5/D     6/E     7/F

U 0A40         11752:  11759:  11760:  11761:  11766:  11767:  11768:  11769:
U 0A48         11770:  11774:  11776:  11777:  11781:  11782:  11783:  11785:
U 0A50         11786:  11792:  11793:  11795:  11796:  11798:  11799:  11800:
U 0A58         11805:  11806:  11807:  11809:  11810:  11814:  11815:  11816:
U 0A60         11817:  11822:  11824:  11825:  11826:  11827:  11832:  11834:
U 0A68         11835:  11836:  11837:  11842:  11843:  11844:  11845:  11846:
U 0A70         11847:  11848:  11849:  11850:  11855:  11856:  11857:  11858:
U 0A78         11859:  11860:  11861:  11862:  11863:  11979:  11981:  11985:
U 0A80         11988:  11991:  11993:  11994:  11995:  11998:  12000:  12001:
U 0A88         12003:  12009:  12010:  12011:  12013:  12014:  12015:  12185:
U 0A90         12186:  12187:  12188:  12189:  12190:  12191:  12193:  12194:
U 0A98         12197:  12198:  12200:  12201:  12202:  12203:  12204:  12205:
U 0AA0         12206:  12211:  12212:  12213:  12214:  12215:  12220:  12225:
U 0AA8         12226:  12227:  12606:  12607:  12608:  12609:  12610:  12611:
U 0AB0         12612:  12614:  12615:  12618:  12619:  12621:  12622:  12623:
U 0AB8         12624:  12625:  12626:  12627:  12630:  12631:  12632:  12633:
U 0AC0         12634:  12635:  12636:  12637:  12639:  12640:  12641:  12643:
U 0AC8         12644:  12645:  12646:  12651:  12652:  12653:  12654:  12655:
U 0AD0         12660:  12661:  12662:  12663:  12664:  12665:  12670.  12671:
U 0AD8         12672:  12673:  12674:  12680:  12681:  12687:  12688:  12689:
U 0AE0         12691:  12692:  12693:  12699:  12700:  12701:  12707:  12708:
U 0AE8         12709:  12710:  12711:  12712:  12717:  12718:  12719:  12720:
U 0AF0         12722:  12723:  12724:  12731:  12732:  12733:  12738:  12739:
U 0AF8         12740:  12741:  12742:  12743:  12744:  12749:  12750:  12751:
U 0B00         12756:  12757:  12758:  12763:  12764:  12765:  12766:  12767:
U 0B08         12768:  12769:  12770:  12772:  12773:  12774:  12776:  12777:
U 0B10         12778:  12779:  12784:  12785:  12786:  12787:  12788:  12793:
U 0B18         12795:  12796:  12797:  12798:  12799:  12802:  12803:  12804:
U 0B20         12805:  12806:  12807:  12808:  12809:  12810:  12811:  12812:
U 0B28         12813:  12814:  12815:  12818:  12819:  12820:  12821:  12822:
U 0B30         12823:  12824:  12825:  12826:  12827:  12828:  12829:  12830:
U 0B38         12897:  12899:  12903:  12906:  12909:  12911:  12912:  12913:
U 0B40         12916:  12918:  12919:  12925:  12926:  12929:  12930:  12931:
U 0B48         12933:  12934:  12935:  12942:  12943:  12944:  12949:  12953:
U 0B50         12954:  12955:  12956:  12957:  12958:  12959:  12963:  12964:
U 0B58         12965:  12966:  12967:  12968:  12969:  12970:  12975:  12977:
U 0B60         12978:  12980:  12981:  12982:  13088:  13090:  13094:  13097:
U 0B68         13100:  13102:  13103:  13104:  13107:  13109:  13110:  13116:
U 0B70         13117:  13118:  13120:  13121:  13122:  13128:  13129:  13130:
U 0B78         13136:  13137:  13138:  13139:  13140:  13145:  13146:  13147:
U 0B80         13148:  13150:  13151:  13152:  13159:  13160:  13161:  13166:
U 0B88         13167:  13168:  13173:  13174:  13175:  13176:  13180:  13181:
U 0B90         13183:  13184:  13191:  13192:  13193:  13194:  13195:  13197:
U 0B98         13198:  13199:  13758:  13759:  13760:  13761:  13762:  13763:
U 0BA0         13764:  13766:  13767:  13770:  13771:  13773:  13774:  13775:
U 0BA8         13776:  13777:  13778:  13779:  13784:  13785:  13786:  13787:
U 0BB0         13788:  13791:  13792:  13793:  13794:  13795:  13796:  13798:
U 0BB8         13799:  13800:  13801:  13803:  13804:  13805:  13806:  13807:
U 0BC0         13812:  13813:  13814:  13815:  13818:  13819:  13821:  13822:
U 0BC8         13828:  13829:  13830:  13831:  13832:  13834:  13835:  13836:
U 0BD0         14383:  14384:  14385:  14386:  14387:  14388:  14390:  14391:
U 0BD8         14394:  14395:  14397:  14398:  14399:  14400:  14401:  14402:
U 0BE0         14403:  14408:  14409:  14410:  14411:  14412:  14415:  14419:
```

```
;Location        0/8     1/9     2/A     3/B     4/C     5/D     6/E     7/F

U 0BE8          14420:  14421:  14422:  14423:  14424:  14425:  14426:  14430:
U 0BF0          14431:  14432:  14433:  14434:  14435:  14436:  14437:  14438:
U 0BF8          14439:  14442:  14443:  14444:  14445:  14446:  14447:  14448:
U 0C00          14452:  14453:  14454:  14455:  14525:  14527:  14531:  14534:
U 0C08          14537:  14539:  14540:  14541:  14544:  14546:  14547:  14553:
U 0C10          14554:  14555:  14557:  14558:  14559:  14565:  14566:  14567:
U 0C18          14573:  14574:  14575:  14576:  14577:  14582:  14583:  14584:
U 0C20          14585:  14587:  14588:  14589:  14596:  14597:  14598:  14603:
U 0C28          14604:  14605:  14610:  14611:  14612:  14613:  14617:  14618:
U 0C30          14620:  14621:  14627:  14628:  14629:  14630:  14631:  14633:
U 0C38          14634:  14635:  14642:  14643:  14644:  14647:  14648:  14649:
U 0C40          14650:  14651:  14652:  14654:  14655:  14656:  14657:  14659:
U 0C48          14660:  14661:  14662:  14663:  14669:  14670:  14671:  14672:
U 0C50          14676:  14677:  14679:  14680:  14686:  14687:  14688:  14690:
U 0C58          14691:  14692:  15270:  15271:  15272:  15273:  15274:  15275:
U 0C60          15276:  15278:  15279:  15282:  15283:  15285:  15286:  15287:
U 0C68          15288:  15289:  15290:  15291:  15296:  15297:  15298:  15299:
U 0C70          15300:  15303:  15304:  15310:  15311:  15312:  15313:  15314:
U 0C78          15315:  15316:  15320:  15321:  15322:  15323:  15326.  15579:
U 0C80          15581:  15585:  15588:  15594:  15595:  15596:  15597:  15598:
U 0C88          15599:  15601:  15603:  15604:  15605:  15608:  15609:  15612:
U 0C90          15613:  15615:  15617:  15619:  15620:  15622:  15623:  15624:
U 0C98          15625:  15626:  15627:  15628:  15629:  15630:  15632:  15633:
U 0CA0          15634:  15635:  15636:  15637:  15638:  15640:  15641:  15643:
U 0CA8          15644:  15646:  15647:  15648:  15649:  15650:  15652:  15655:
U 0CB0          15656:  15657:  15659:  15662:  15663:  15666:  15667:  15669:
U 0CB8          15670:  15671:  15672:  15863:  15864:  15865:  15870:  15871:
U 0CC0          15872:  15873:  15874:  15875:  15876:  15877:  15881:  15882:
U 0CC8          15883:  15884:  15885:  15886:  15887:  15888:  15889:  15890:
U 0CD0          15894:  15896:  15897:  15899:  15901:  15902:  15904:  15905:
U 0CD8          15907:  15908:  15909:  15911:  15912:  15913:  15916:  15919:
U 0CE0          15922:  15924:  15925:  15930:  15931:  15932:  15933:  15934:
U 0CE8          15935:  15936:  15937:  15938:  15939:  15940:  15941:  15946:
U 0CF0          15947:  15948:  16231:  16232:  16233:  16238:  16239:  16240:
U 0CF8          16241:  16242:  16243:  16245:  16246:  16247:  16436:  16437:
U 0D00          16438:  16441:  16442:  16443:  16446:  16447:  16448:  16456:
U 0D08          16457:  16458:  16463:  16464:  16465:  16466:  16467:  16468:
U 0D10          16469:  16470:  16471:  16472:  16473:  16474:  16485:  16486:
U 0D18          16487:  16488:  16491:  16492:  16493:  16494:  16495:  16950:
U 0D20          16951:  16952:  16955:  16956:  16957:  16961:  16962:  16963:
U 0D28          16965:  16966:  16967:  16968:  16970:  16971:  16972:  16973:
U 0D30          16974:  16976:  16977:  16983:  16984:  16986:  16987:  16988:
U 0D38          16989:  16990:  16995:  16996:  16998:  16999:  17001:  17002:
U 0D40          17003:  17004:  17005:  17006:  17007:  17012:  17013:  17014:
U 0D48          17021:  17022:  17023:  17026:  17028:  17033:  17034:  17035:
U 0D50          17036:  17037:  17038:  17039:  17040:  17041:  17043:  17046:
U 0D58          17048:  17050:  17051:  17059:  17060:  17061:  17062:  17063:
U 0D60          17064:  17071:  17072:  17073:  17076:  17081:  17082:  17083:
U 0D68          17084:  17086:  17090:  17091:  17093:  17094:  17095:  17101:
U 0D70          17102:  17103:  17106:  17107:  17108:  17111:  17112:  17113:
U 0D78          17119:  17120:  17122:  17123:  17126:  17127:  17128:  17129:
U 0D80          17130:  17132:  17133:  17135:  17138:  17140:  17141:  17142:
U 0D88          17145:  17146:  17149:  17150:  17151:  17153:  17154:  17155:
```

```
                                        Location / Line Number Index

;Location          0/8       1/9       2/A       3/B       4/C       5/D       6/E       7/F

U 0D90            17156:    17157:    17158:    17160:    17161:    17162:    17163:    17164:
U 0D98            17165:    17166:    17170:    17172:    17173:    17175:    17178:    17179:
U 0DA0            17180:    17181:    17183:    17184:    17186:    17187:    17189:    17190:
U 0DA8            17191:    17192:    17193:    17194:    17195:    17196:    17198:    17201:
U 0DB0            17203:    17208:    17209:    17210:    17211:    17212:    17214:    17217:
U 0DB8            17311:    17313:    17317:    17319:    17322:    17324:    17325:    17326:
U 0DC0            17329:    17330:    17331:    17340:    17341:    17342:    17349:    17350:
U 0DC8            17351:    17354:    17355:    17356:    17357:    17358:    17360:    17361:
U 0DD0            17362:    17364:    17365:    17366:    17367:    17368:    17369:    17372:
U 0DD8            17373:    17374:    17375:    17376:    17377:    17378:    17379:    17380:
U 0DE0            17386:    17387:    17390:    17391:    17392:    17393:    17394:    17396:
U 0DE8            17397:    17398:    17400:    17401:    17402:    17403:    17404:    17405:
U 0DF0            17408:    17409:    17410:    17411:    17412:    17413:    17414:    17417:
U 0DF8            17444:    17446:    17450:    17453:    17456:    17459:    17460:    17461:
U 0E00            17466:    17467:    17468:    17475:    17476:    17477:    17483:    17484:
U 0E08            17486:    17488:
```

```
          Words not
          in bounds
    ENKCG       512

Used            512
Remaining

Total microwords used in memory C: 512
Total microwords remaining in memory C: 0
Highest address used in memory C: 180 (hex)
```

```
           Words not
           in bounds
    ENKCG       2644

Used            2644
Remaining

Total microwords used in memory U: 2644
Total microwords remaining in memory U: 0
Highest address used in memory U: 0E09 (hex)
```

```
; The file used in this assembly is:
ENKCG.MIC

Pass 1 warnings:     0   Pass 2 warnings:      0
        Unsplit Binary Lines:    0
Pass 1 errors:       0   Pass 2 errors:        0
```